

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-093523
 (43)Date of publication of application : 04.04.1997

(51)Int.Cl. H04N 5/765
 H04N 5/781
 G06F 12/00
 G11C 16/06
 H04N 5/907

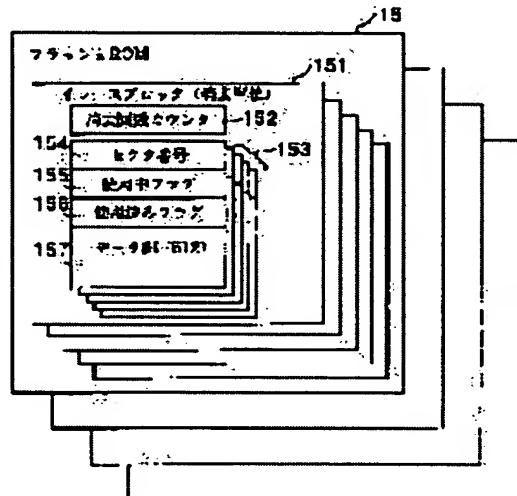
(21)Application number : 07-251241 (71)Applicant : CANON INC
 (22)Date of filing : 28.09.1995 (72)Inventor : OGAWA TAKESHI

(54) ELECTRONIC CAMERA

(57)Abstract:

PROBLEM TO BE SOLVED: To make a flush ROM with a large delete unit adaptive to a file system.

SOLUTION: Plural data areas 157 and management areas (consisting of a sector number 154, a busy flag 155 and a used flag 156) corresponding to the data areas are formed to a flush memory ROM mounted onto the electronic camera and the memory is managed. In the case of data write, a write command attended with a logical sector number representing a data write destination is received to write the logic sector number to a corresponding management area. Furthermore, in the case of reading data, a read command attended with a logical sector number representing the data read source is received to retrieve the management area to which the logic sector number is stored and data stored in a data area corresponding to the retrieved management area are read out.



LEGAL STATUS

[Date of request for examination] 27.09.2002

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3727983

[Date of registration] 07.10.2005

[Number of appeal against examiner's
decision of rejection]

[Date of requesting appeal against
examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平9-93523

(43)公開日 平成9年(1997)4月4日

| (51)Int.Cl. ⁶ | 識別記号 | 序内整理番号 | F I | 技術表示箇所 |
|--------------------------|-------|--------|--------------|---------|
| H 04 N 5/765 | | | H 04 N 5/781 | 5 1 0 E |
| | 5/781 | | G 06 F 12/00 | 5 0 1 A |
| G 06 F 12/00 | 5 0 1 | | H 04 N 5/907 | B |
| G 11 C 16/06 | | | G 11 C 17/00 | 3 0 9 F |
| H 04 N 5/907 | | | | |

審査請求 未請求 請求項の数7 O L (全32頁)

(21)出願番号 特願平7-251241

(22)出願日 平成7年(1995)9月28日

(71)出願人 000001007

キヤノン株式会社

東京都大田区下丸子3丁目30番2号

(72)発明者 小川 武志

東京都大田区下丸子3丁目30番2号 キヤ
ノン株式会社内

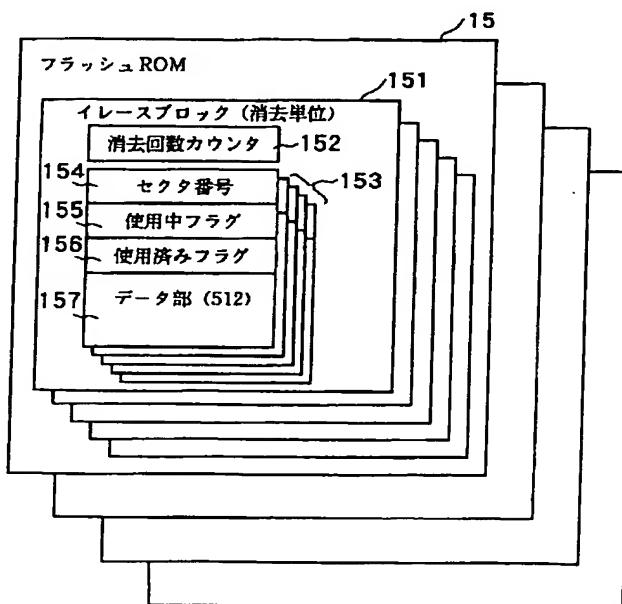
(74)代理人 弁理士 大塚 康徳 (外1名)

(54)【発明の名称】 電子カメラ

(57)【要約】

【課題】消去単位の大きいフラッシュROMをファイルシステムと適合させることを可能とした電子カメラを提供する。

【解決手段】電子カメラに搭載されたフラッシュROMに、複数のデータ部157、及び各データ部に対応する管理領域（セクタ番号154、使用中フラグ155、使用済フラグ156で構成される）を形成して記憶管理を行う。データの書き込みにおいては、データの書き込み先を示す論理セクタ番号を伴う書き込み指示を受けて、複数のデータ部のうちの一つにデータを書き込むと共に、対応する管理領域にその論理セクタ番号を書き込む。また、データの読み出し時は、データの読み出し元を示す論理セクタ番号を伴う読み出し指示を受けて、該論理セクタ番号が格納された管理領域を検索し、検索された管理領域に対応するデータ部に格納されたデータを読み出す。



【特許請求の範囲】

【請求項 1】 データをフラッシュ ROM に保存する電子カメラであつて、前記フラッシュ ROM に形成された、複数のデータ領域、及び各データ領域に対応する管理領域と、データの書き込み先を示す指定情報を伴う書き込み指示を受けて、前記複数のデータ領域のうちの一つのデータ領域にデータを書き込むと共に、該指定情報を該データ領域に対応する管理領域に書き込む書き込み手段と、データの読み出し元を示す指定情報を伴う読み出し指示を受けて、該指定情報が格納された管理領域を検索し、検索された管理領域に対応するデータ領域に格納されたデータを読み出す読み出し手段とを備えることを特徴とする電子カメラ。

【請求項 2】 前記管理領域は、対応するデータ領域に書き込みが可能であることを示す未使用状態と、データ領域に書き込まれたデータが有効であることを示す使用中状態と、データ領域に書き込まれたデータが無効であることを示す使用済状態の少なくとも 3 通りの状態のいずれかを示す状態情報を格納し、前記書き込み手段は、状態情報が未使用状態となっている管理領域を検索して検索された管理領域及びこれに対応するデータ領域に前記指定情報及びデータを書き込み、該管理領域の状態情報を使用中に変更することを特徴とする請求項 1 に記載の電子カメラ。

【請求項 3】 前記書き込み手段は、前記書き込み指示に伴う指定情報を有する管理領域を検索し、検索された管理領域の状態情報を使用済状態に変更することを特徴とする請求項 2 に記載の電子カメラ。

【請求項 4】 フラッシュ ROM 内の消去単位である消去ブロックにおいて、状態情報が使用中状態である全ての管理領域を検索し、検索された管理領域及び対応するデータ領域の内容を当該消去ブロック外へ移動させる移動手段と、

前記移動手段の実行の後に前記消去ブロックの消去動作を行う消去手段とを更に備えることを特徴とする請求項 2 に記載の電子カメラ。

【請求項 5】 セルフタイマー撮影を行うためのタイマー手段を更に備え、

前記移動手段及び消去手段の実行を前記タイマー手段による時間の計測中に実行することを特徴とする請求項 4 に記載の電子カメラ。

【請求項 6】 電源の使用を管理する電源管理手段を更に備え、

前記電源管理手段は、ストロボ充電、機構部分の駆動あるいは CCD 駆動の少なくともいずれかに用いられる電源の電力を前記フラッシュ ROM への書き込みアクセスに割り振ることを特徴とする請求項 1 乃至 5 のいずれかに記載の電子カメラ。

【請求項 7】 前記電源管理手段は、前記電源による電

力の供給を、前記ストロボ充電、機構部分の駆動、CCD の駆動、前記フラッシュ ROM の書き込み処理に対して時分割に割り当てるなどを特徴とする請求項 6 に記載の電子カメラ。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】 本発明は、フラッシュ ROM へデータ記憶が可能な電子カメラに関する。

【0002】

【従来の技術】 電子カメラにおいて、撮影して得られた画像データは内蔵する DRAM、あるいは外付けのメモリカード等の記憶媒体に保存するが、これらの記憶媒体は電力供給を遮断すると記憶内容が消滅してしまうため、電池などで常時バックアップする必要がある。また、記憶内容の保持に電気的なバックアップが不要な記憶媒体としては、磁気ディスクが挙げられるが、磁気ディスク装置は比較的大きいため、電子カメラの携帯性を損なうことになる。また、記憶内容の保持に電気的のバックアップが不要な記憶媒体として、書き換えが可能な ROM、即ちフラッシュ ROM がある。

【0003】 フラッシュ ROM は現在いろいろなタイプのものがあるが大きく分けてフラッシュ DISK 用に開発されたタイプとパーソナルコンピュータの BIOS 用に開発されたものがある。

【0004】 前者は消去単位がハードディスクで一般的な 512 バイトであり、ファイルシステムとの整合性が非常に良い。後者のフラッシュ ROM は消去単位が例えば 64K などといった大きなブロック単位でしか行えない様になっている。また、 PROM の様に書き込み電圧として 12V 等の電圧が必要なものもある。後者のフラッシュ ROM の方が安価に入手できるがファイルシステムとの整合性が悪い為に特に小容量の記録メディアとしては使う事が出来なかった。

【0005】

【発明が解決しようとする課題】 以上のように、BIOS 用に設計されたフラッシュ ROM は、その消去単位が大きく、ファイルシステムとの整合性が悪いが、安価で入手しやすい。従って、そのようなフラッシュ ROM をファイルシステムに適用できれば、記憶保持用の電源が不要で、安価で小容量、小型の記録メディアを提供することができ、電子カメラの記憶媒体としては都合が良い。

【0006】 本発明は、上記問題に鑑みてなされたものであり、消去単位の大きいフラッシュ ROM をファイルシステムと適合させることを可能とした電子カメラを提供することを目的とする。

【0007】

【課題を解決するための手段】 上記の目的を達成するための本発明の電子カメラは以下の構成を備える。即ち、データをフラッシュ ROM に保存する電子カメラであつ

て、前記フラッシュROMに形成された、複数のデータ領域、及び各データ領域に対応する管理領域と、データの書き込み先を示す指定情報を伴う書き込み指示を受けて、前記複数のデータ領域のうちの一つのデータ領域にデータを書き込むと共に、該指定情報を該データ領域に対応する管理領域に書き込む書き込み手段と、データの読み出し元を示す指定情報を伴う読み出し指示を受けて、該指定情報が格納された管理領域を検索し、検索された管理領域に対応するデータ領域に格納されたデータを読み出す読み出し手段とを備える。

【0008】また、好ましくは、前記管理領域は、対応するデータ領域に書き込みが可能であることを示す未使用状態と、データ領域に書き込まれたデータが有効であることを示す使用中状態と、データ領域に書き込まれたデータが無効であることを示す使用済状態の少なくとも3通りの状態のいずれかを示す状態情報を格納し、前記書き込み手段は、状態情報が未使用状態となっている管理領域を検索して検索された管理領域及びこれに対応するデータ領域に前記指定情報及びデータを書き込み、該管理領域の状態情報を使用中に変更する。各データ領域について使用中、使用済、未使用の3状態を示すことにより、状態情報をチェックすることでデータの書き込みが可能な領域、有効なデータが存在する領域を把握でき、電子カメラにおけるフラッシュROMの記憶管理をより適切に行えるからであるまた、好ましくは、前記書き込み手段は、前記書き込み指示に伴う指定情報を有する管理領域を検索し、検索された管理領域の状態情報を使用済状態に変更する。ある指定情報で特定されるデータ領域の内容を更新した場合に、更新前のデータ（即ち不要なデータ）であることを明示でき、電子カメラにおけるフラッシュROMの適切な記憶管理が行えるからである。

【0009】また、好ましくは、フラッシュROM内の消去単位である消去ブロックにおいて、状態情報が使用中状態である全ての管理領域を検索し、検索された管理領域及び対応するデータ領域の内容を当該消去ブロック外へ移動させる移動手段と、前記移動手段の実行の後に前記消去ブロックの消去動作を行う消去手段とを更に備える。データ更新等によって無効データとなった使用済状態のデータ領域を、書き込みが可能な未使用状態とすることが可能となり、電子カメラにおけるフラッシュROMの利用効率が向上するからである。

【0010】また、好ましくは、セルフタイマー撮影を行うためのタイマーハンドルを更に備え、前記移動手段及び消去手段の実行を前記タイマーハンドルによる時間の計測中に実行する。セルフタイマー撮影中に、移動手段、消去手段を実行することにより電子カメラの操作に支障なくフラッシュROMの整理を行うことが可能となるからである。

【0011】また、好ましくは、電源の使用を管理する電源管理手段を更に備え、前記電源管理手段は、ストロ

ボ充電、機構部分の駆動あるいはCCD駆動の少なくともいずれかに用いられる電源の電力を前記フラッシュROMへの書き込みアクセスに割り振る。すると専用電源や、機構部分の駆動、CCD駆動等に用いられる電源を、フラッシュROMの書き込みアクセスに用いることができ、フラッシュROMへの書き込み専用の電源を用意する必要がなくなるからである。

【0012】前記電源管理手段は、前記電源による電力の供給を、前記ストロボ充電、機構部分の駆動、CCDの駆動、前記フラッシュROMの書き込み処理に対して時分割に割り当てる。時分割で電源を割り当てるので、短い保留時間の後に必ず電源を確保できるようになり、各種動作の処理がスムーズに実行される。

【0013】

【発明の実施の形態】以下に添付の図面を参照して本発明の実施の形態を説明する。

【0014】【実施形態1】

<カメラシステムの構成>図1は実施形態1におけるカメラシステムの構成を表すブロック図である。本カメラシステムは、電子カメラと、これに着脱可能な外部記憶媒体17、PC通信インターフェース19、及びPC通信インターフェース19を介して電子カメラと通信可能に接続されたパーソナルコンピュータ22から構成される。

【0015】1はレンズであり、2はレンズ1を通った光を電気信号として出力するCCDユニットである。3はA/Dコンバータであり、CCDユニット2からのアナログ信号をデジタル信号へ変換する。4はSSGユニットであり、CCDユニット2とA/Dコンバータ3に同期信号を供給する。5はCPUであり、本カメラシステムにおける各種の制御を実現する。

【0016】6は信号処理アクセラレータであり、信号処理を高速に実現する。7は電池であり、8は、電池7よりの電力を電子カメラ全体へ供給するためのDC/DCコンバータである。9は電源コントローラユニットであり、DC/DCコンバータ8をコントロールする。10はパネル操作・表示装置・電源のコントロールを行うマイクロコンピュータである。11はユーザへ各種の情報を表示する表示装置であり、液晶パネル等が用いられる。12はコントロールパネルであり、ユーザが直接操作するレリーズスイッチを含む。

【0017】13はROMであり、OS等のシステムプログラムを格納する。14はDRAMであり、本電子カメラの主記憶である。15はフラッシュROMであり、内蔵記憶媒体として使用する。16はPCMCIAカードのインターフェース部、17はATAハードディスクなどの外部記憶媒体、18は拡張バスインターフェースである。19はPC通信インターフェースであり、パーソナルコンピュータ等を接続してデータの授受を行う。20はDMAコントローラ、21はストロボである。また、2

2はパーソナルコンピュータであり、PC通信インターフェース19を介して、電子カメラとの通信を行う。

【0018】<撮影動作>この電子カメラの撮影時の動作を簡単に説明する。コントロールパネル12のレリーズスイッチをユーザが押すと、CPU5がそのことを検出して撮影シーケンスを開始する。以下の動作は全てCPU5によるコントロールで行われることを前提とする。

【0019】さて、レリーズスイッチの押下により、SSG4がCCD2を駆動する。CCD2から出力されるアナログ信号は、A/Dコンバータ3でデジタル信号へ変換される。A/Dコンバータ3の出力は、DMAコントローラ20によってDRAM14へDMA転送される。1フレーム分のDMA転送が終了した時点でCPU5は、信号処理シーケンスを開始する。

【0020】信号処理シーケンスでは、フラッシュROM15から信号処理プログラムを主記憶(DRAM14)上に読み出し、主記憶上のデータを信号処理アクセラレータ6へ転送し信号処理を行う。但し、信号処理アクセラレータ6は信号処理の全てを行うわけではなく、CPU5で行う処理の特に時間のかかる処理などを助ける演算回路であり、CPU5の処理ソフトウェアと連携して動作する。信号処理の一部または全部が終了すると画像ファイルとしてフラッシュROM15へ記録する。この時記録するファイルフォーマットが圧縮処理を必用とするのであれば圧縮も行う。

【0021】信号処理プログラムは、フラッシュROM15の中でファイルシステムが管理するファイルの1つである。カメラのプログラムはOSやファイルシステムといっしょにROM13に納められている。カメラのプログラムは、特定のファイル名のファイルをプログラムであると認識する。

【0022】フラッシュROM15の中でファイルは不連続に配置されている上に、本実施形態のファイルシステムが頻繁に再配置を行うため、フラッシュROM15内の制御プログラムをCPUが直接実行することはできない。従って、主記憶(DRAM14)に読み出して実行させなければならない。更に、主記憶はメモリマネージャが動的に記憶場所をアロケーションするため、特定のアドレスに格納されることを想定したソフトウェアであってはならない。そのため、本実施形態で信号処理を行うプログラムのファイルは図41のような形式となっている。

【0023】図41は本実施形態におけるフラッシュROMへの制御プログラムの格納状態を説明する図である。図41において、識別コードはファイルがプログラムであることを確認するためのコードである。ファイルは可変長のレコードの集合として表現されている。レコードには、始めに当該レコードに格納されている情報の種類を識別するIDがあり、次にそのレコードの大きさ

を示す値が格納されている。

【0024】そして、プログラムのレコードとリレーション情報のレコードがファイルに格納されている。プログラムコードは例えば図42のようなデータである。図42は相対アドレスで表現されたプログラムコードの一例を表す図である。図42では、0050番地にジャンプ命令があるが、CPUはこの命令を絶対番地へのジャンプ命令と認識する。この命令のオペランドは相対アドレスで表現されている。

【0025】図42のリレーション情報レコードのデータは図43のような形式で納められている。即ち、図42のプログラムの中で、絶対番地へ変換しなければならないデータ(相対アドレス表現になっているデータ)のプログラム番地を示すアドレステーブルがリレーション情報として格納される。

【0026】図41のファイルを主記憶にロードするための領域を確保すると、ROM13のOSのメモリマネージャがアドレスを決定する。メモリマネージャのアロケーションはC言語ではalloc関数に相当する機能である。メモリマネージャがプログラム用に8710番地を割り当てた場合図44のようにプログラムがロードされる。図44は図41のプログラムを主記憶の8710番地へマッピングした場合のプログラムコードを示す図である。ジャンプ命令のオペランドが実際の絶対番地に置き換えられている。この実アドレスへの変換をしながらプログラムを主記憶へ読み出すと言う作業を行うプログラムはROM13に格納されている。

【0027】以上のように構成することにより、記憶媒体に信号処理ソフトウェアや圧縮ソフトウェアをファイル形式で格納することができる。その結果、カメラが最終ユーザの元へ届いてから、新しい信号処理アルゴリズムや、Windows(商標)のBMP形式やTIFF形式、あるいは将来新たに登場する形式等、多種多様なファイル形式への対応が可能となる。

【0028】以上の様に、本実施形態1における電子カメラは、撮影画像をフラッシュROM15へファイルするものである。

【0029】<デバイスドライバインターフェース>図2は、本実施形態の電子カメラにおけるファイルシステムの階層構造を表す図である。最上位の層がユーザアプリケーション101である。ユーザアプリケーション101は電子カメラの内部で動くソフトウェアであり、ファイルをファイル名でオープンして読み書きした後クローズする。

【0030】ユーザアプリケーション101から直接ファンクションコールによって呼び出されるのがファイルシステムAPI層102である。このファイルシステムAPI層102がドライブ名とファイルシステムを関連付けて管理している。各ドライブ毎にファイルシステムアーキテクチャ層をマウントする用に構成しているた

め、複数のファイルシステムアーキテクチャを混在させる事が可能となっている。

【0031】ファイルシステムアーキテクチャ層103が実際のファイル管理を行う部分である。最下位の層がブロックデバイス層104である。ファイルシステムアーキテクチャ層103がブロックデバイス層104の提供するサービスを利用してファイル入出力を実現している。このブロックデバイス層104では、データをセクタという単位で管理しており、1セクタは例えば512バイトである。このブロックデバイス層104でデバイスごとの入出力制御の違いと、ヘッドやシリンドなどパラメータの違いを吸収している。このように構成しているため、同時に複数の種類のデバイスを混在させることができる。

【0032】本実施形態の電子カメラでは、特にブロックデバイス層104におけるフラッシュROMの記憶管理方法に特徴を有する。

【0033】図1で示したフラッシュROM11には、現在いろいろなタイプのものがあるが、大きく分けてフラッシュDISK用に開発されたタイプとパソコンコンピュータのBIOS用に開発されたタイプがある。前者は消去単位がハードディスクで一般的な512バイトであり、ファイルシステムとの整合性が非常に良い。後者のフラッシュROMは消去単位が例えば64kなどといった大きなブロック単位でしか行えない様になっている。また、PROMの様に書き込み電圧として12V等の電圧が必要なものもある。しかしながら、後者のタイプのフラッシュROMは安価で入手が容易である。本実施形態では、後者の様な特徴を持つフラッシュROMでありながらファイルシステムに対してハードディスク同様のサービスを提供する。

【0034】<フラッシュROMドライバインターフェース>一般的にブロックデバイスがファイルシステムへ提供するサービスは以下の2つである。即ち、

(1) ロジカルセクタナンバーで指定したセクタからの読み出し

(2) ロジカルセクタナンバーで指定したセクタからの書き込みである。そして、これに加えて

(3) ロジカルセクタナンバーで指定したセクタの開放の機能があれば、フラッシュROMのドライバは必要に応じて不要なセクタを消去することが可能となるため、効率良くフラッシュROMを消去することができる。

【0035】(3)に挙げた機能は、通常のDISKでは必要のない機能だが、キャッシングを持ったシステムだと、積極的にキャッシングリストから削除できるので、結果的にキャッシングのヒット率を上げる効果がある。ファイルシステムは、ファイルの消去等で不必要となったセクタを(3)の機能を用いてデバイスドライバへ通知する。フラッシュROMの消去は非常に時間がかかる処理だが、CPU時間をほとんど消費しないためバックグラ

ンド処理で行うのが良い。

【0036】後述の<FATキャッシュ>においても説明するが、本実施形態のキャッシングは、新しいデータ(キャッシング上に無いデータ)をアクセスする場合にキャッシングリストの中で最も古いデータを廃棄する。不要セクタをキャッシングリストの最後へ移動させる(即ち、最も古くアクセスしたデータがキャッシングの後ろへ移動する)ことで有効なデータがキャッシングから廃棄される可能性が低くなる。特にコンパイラ等の中間ファイルを多く生成するシステムでは、消去すべき中間ファイルがキャッシングに残っている可能性が高く、上記のキャッシング管理はヒット率の向上に非常に有効である。

【0037】図3は、デバイスドライバの管理プロックをC言語で記述した宣言文を示す図である。構造体のNextは、次のデバイスへのリングポインタであり、メモリ中のデバイスを検索する目的で使用される。DevNameは、デバイスの名前として使用される。InitDevは、デバイスの初期化ルーチンへのポインタである。Shutdownは、デバイスのシャットダウンルーチンへのポインタである。ReadSectorは、ロジカルセクタを指定して媒体の内容をバッファへ転送するルーチンへのポインタである。WriteSectorは、ロジカルセクタを指定してバッファの内容を媒体へ転送する(書き込む)プログラムへのポインタである。ReleaseSectorはロジカルセクタを指定して、セクタを解放するルーチンへのポインタである。

【0038】ファイルシステムは、この構造体を仲介してデバイスドライバを利用することになる。固定ディスクやフロッピーディスクの場合、ReleaseSectorには何も仕事をしないプログラムへのポインタが代入されている。または、ディスクキャッシングのキャッシングリストから指定セクタを削除するポインタでもよい。

【0039】<フラッシュROM管理方法>フラッシュROMに対するデータ書き込みは、上位層のファイルシステムからセクタ単位で行われる。図4は、フラッシュROM上のセクタ構造の例を示す図である。図4において、151はイレースブロックである。このイレースブロック151は消去の単位であり、フラッシュROMの技術用語ではセクタと呼ばれるものである。しかしながら、ファイルシステムが扱う単位である“論理セクタ”と区別する為に、ここではイレースブロックと呼ぶことにする。

【0040】図4によれば、システム中に複数のフラッシュROM15が搭載されていて、各フラッシュROM15は複数のイレースブロック151によって構成される。更に、各イレースブロック151は消去回数カウンタ152と複数のセクタ153によって構成されている様子をあらわしている。消去回数カウンタ152は、イ

レースブロック 151 を消去した回数をカウントする為に用いる部分である。各セクタ 153 は、管理領域とデータ領域とを有する。管理領域は、論理セクタ番号を表すセクタ番号 154 と、セクタが有効利用されているかどうかを表わす使用中フラグ 155 と、セクタとしての利用が終了したことを表わす使用済みフラグ 156 とで構成される。また、データ領域は、512 バイトのデータ部 157 によって構成されている。

【0041】データ領域と管理領域は、隣り合って配置する必要は無く、図5の様にまとめて管理することも考えられる。図5は、管理領域用データと、データ領域とを分離して格納する構成を表す図である。セクタ番号テーブルには、複数のセクタ 153 の各セクタ番号 154 が格納される。また、フラグテーブルには、使用中フラグ 155 、使用済みフラグ 156 が格納される。更に、データテーブルには、データ部 157 の内容が格納される。以上のようなデータ構成をとることも可能であるが、少なくとも管理領域と、これに対応するデータ領域とは同じイレースブロック内に納めるのが好ましい。

【0042】なお、システムは「使用中フラグ 155」より「使用済みフラグ 156」の方を優先的に評価する。図6は、各フラグの状態に対応した意味を示す図である。図中、FALSEは、消去後の状態と同じ値とする。使用中フラグ 155 が TRUE であっても、使用済みフラグ 156 が FALSE であれば、当該セクタのデータは無効である。

【0043】<フラッシュROMの論理セクタ書き換え>フラッシュROMはPROM同様、データを書き換える為に、一度消去してから再書き込みをしなければならない。しかも、消去の最少単位が大きく（例えば64kバイト）消去時間が長い（例えば1秒）。そこで上位層のファイルシステムが、特定のセクタを書き換えようとした場合、消去済みの領域へ論理セクタを移動させることで、消去動作をせずに、見かけ上で論理セクタのデータ書き換えを実現する。

【0044】図7はセクタの書き換え手順を説明する図である。同図を用いて、8番セクタ（論理セクタ番号が8のセクタ）の書き換えを例にして詳しく説明する。図7中、左側が書き換え前の状態であり（(a)の状態）、右側が書き換え後の状態（(b)の状態）である。また、図7において、管理領域中の数字は論理セクタ番号を表し、（使用中）は使用中フラグ 155 が TRUE で使用済みフラグ 156 が FALSE の状態、（使用済）は使用中フラグ 155 と使用済みフラグ 156 と共に TRUE の状態を示す。

【0045】“セクタ番号8（使用中）”の場所に、8番セクタのデータが格納されている。今、8番セクタが FAT やファイルの一部として利用されていて、その内容を変更したい場合に上位層から 8番セクタの書き換え要求が発生したとする。書き換え要求が発生すると、フ

ラッシュROMのデバイスドライバは、フラッシュROMの未使用セクタを検索し、その場所を新たな8番セクタの場所としてセクタ番号と更新後のデータを格納し、使用中フラグを TRUE にする。次に、以前8番セクタだったセクタの使用済みフラグを TRUE にする。このような手順で8番セクタのデータの書き換えが実現される。

【0046】<ガベージコレクション>以上の様な方法で論理セクタの書き換えを実行していくと、いずれフラッシュROMのほとんどの領域を“使用済セクタ”にしてしまうことになる。そこであるタイミングでフラッシュROMを一旦消去して“使用済セクタ”を“未使用セクタ”へ戻す必要がある。基本的なガベージコレクションの動作を図8を用いて説明する。図8は、本実施形態におけるフラッシュROMのガベージコレクション動作を説明する図である。

【0047】図中(A)は、ガベージコレクション前の状態である。説明を簡単にするために、本例のフラッシュROMはセクタ6個分の大きさのイレースブロックで構成されているものとする。イレースブロック(1)には使用済セクタが3個と使用中セクタが3個あり、消去回数は5回である（消去回数カウンタ 152 の内容が5である）。イレースブロック(2)には使用中セクタが1個、使用済セクタが1個、未使用セクタが4個あり、消去回数は9回である。この状態からガベージコレクションを開始する。

【0048】先ず、調整対象イレースブロックを選定する。ここで調整対象イレースブロックは消去を行う対象としてイレースブロックとなる。整理対象イレースブロックの選定は、使用済セクタをたくさん含むイレースブロックから優先的に選択すると整理効率が良い。しかし使用済セクタを含まない場合を別として消去回数の少ないイレースブロックを優先的に整理対象とする方法を取ればチップ内のイレースブロックを平均的に使用することができ、書き換え耐久を分散させることができる。選択手順の詳細についてはフローチャートを用いて後述する。

【0049】今、イレースブロック(1)が整理対象として選定されたとする。次に、整理対象であるイレースブロック(1)の使用中セクタ（使用中フラグが TRUE で、使用済みフラグが FALSE のセクタ）を他のイレースブロックに移動させる。使用中セクタの移動手順は、セクタの書き換え時と同様に、他のイレースブロック中の未使用セクタを検索して、使用中セクタ内のデータ領域と管理領域の内容をコピーし、移動元の使用中セクタの使用済みフラグを TRUE にする。なお、未使用セクタが無い場合の処理は、後で述べる。

【0050】図8の(B)は、イレースブロック(1)の使用中セクタをすべてイレースブロック(2)へ移動させた状態である。この結果、イレースブロック(1)

には、使用済セクタしか存在しないことになる。

【0051】次に、使用済セクタだけで構成されているイーストブロックを検索する。ここで、検索を行うのは、通常の書き換え動作の際に偶然イーストブロック内のセクタが全て使用済セクタとなっている場合があるからである。統いて検索されたイーストブロックに対して消去を行う。消去には時間がかかるが、複数のイーストブロックを同時に消去できるため、できるだけ一度に複数のイーストブロックを消去するのが良い。消去が終了すると消去回数カウンタへ消去前の値+1したものと書く。これでガベージコレクション完了である。

【0052】図8の(C)がガベージコレクション終了時の状態である。イーストブロックができるだけ同時に消去した方が効率が良いため、使用済セクタと未使用セクタがある限りたくさんのイーストブロックを同時に調整すると良い。極端に消去回数カウンタの値が他のイーストブロックより少ないものがあれば、使用済セクタを含んでいなくても整理さえすれば、書き換え耐久の分散を図れる。また、一度整理するとデータの配列が変わる為、書き換え耐久分散のきっかけとなる。

【0053】<未使用セクタがない場合>次にシステム中に使用済セクタが有るにもかかわらず未使用セクタが全く無くなってしまった場合のガベージコレクション手順を図9を使って説明する。図9は、未使用セクタが存在しない場合のガベージコレクションの動作を説明する図である。

【0054】先ず、上述した基本的なガベージコレクション手順に従い、イーストブロック(1)を整理対象として選択する。次にイーストブロック(1)の使用中セクタを移動する為の未使用セクタを検索する。未使用セクタがある場合は、上述の基本的ガベージコレクションと同様にセクタの移動を行う。

【0055】一方、検索の結果、未使用セクタが無ければ、DRAM14のヒープエリアからデータの退避に必要な大きさのメモリブロックをアロケーションする。そして調整対象イーストブロック内の使用中セクタをDRAM14へコピーする。この場合は、フラッシュROMの別の領域へセクタを移動する場合と違い、元のセクタの使用済フラグをTRUEにしない。なぜならこの時点で電子カメラの電池7が外れるなどの事故が起った場合に、DRAM内のデータが消滅してしまい、データの修復ができなくなるからである。図9の(B)は、DRAM14の領域へコピーされたセクタを表現している。調整対象のイーストブロックの使用中セクタをすべて退避出来たら、その調整対象のイーストブロックを選択して消去する(図9の(C)参照)。消去が終わった後は未使用セクタがたくさん出来ているはずである。よって、次に未使用領域を検索してDRAM14へ待避してあったデータを復元する。図9の(D)は、ガベージコレクションが完了した状態を示している。

【0056】上記の手順でガベージコレクションをした場合でも、イーストブロックを消去してからデータを復元するまでの間に電子カメラの電池7が外れるなどの事故が起つたらデータの修復をすることはできない。つまり、セクタのデータをDRAM14に退避する方法はできる限り取らない方が、よりシステムの安全性を保つことができる。一回、DRAMを使ってガベージコレクションを行えば未使用セクタができる。したがって、一度DRAM14を使ったガベージコレクションを行い、その後に通常のガベージコレクションを行えば、消去時間は余分にかかるが安全性を高めることができる。逆にDRAM14への退避を積極的に行う(例えばヒープ領域がある限り退避する)と、同時に整理できるイーストブロックが増える為効率を上げることができる。従つて、安全性と効率のどちらを優先するかを指定できるように構成してもよい。またシステムの電源が電池7より供給されている場合は安全性優先、ACアダプタから供給されている場合は効率優先に自動的に切り替わるように構成してもよい。本処理については図36を参照して後述する。

【0057】<イーストブロックを余分に1つ用意>残り容量が極端に少なくなるとガベージコレクションが多発してシステムのパフォーマンスが極端に落ちる。総論理セクタ分を格納できるイーストブロック数よりも1つだけ余分にイーストブロックを使用すれば、そのような事態を避けることが可能である。仮に、1イーストブロックあたり127セクタ格納できるとして、全てのセクタが使用中となった場合、同じ1セクタを10回書き換える場合を例にすると、余分イーストブロックがなければ、10回の消去と1270セクタの書き込みが発生する。しかし、イーストブロックを余分に1つ用意しておけば、10セクタの書き込みしか発生しない。

【0058】よって、本実施形態では、イーストブロックの数はチップの構成で決まるので、最低1つのイーストブロックが余る様な総論理セクタ数を設計する。

【0059】<ガベージコレクションのタイミング>ガベージコレクションは消去動作を伴うために非常に時間がかかる。そのためガベージコレクションをいつ行うかによってカメラの使い勝手を左右することとなる。例えば、セルフタイマなどの数秒間撮影しなくとも良い時にガベージコレクションを行えばユーザがストレスを感じることがない。

【0060】<RAM上の記憶場所管理>フラッシュROM15上ではセクタ番号と実際の記憶場所が関連していないために特定のセクタを読み書きする為にフラッシュROM15を検索しなければならない。そこでシステムがリブートする際に、フラッシュROM15における各セクタの格納アドレスを示す記憶場所管理テーブルをDRAM114上に作成しておくと、フラッシュROM15に対して高速なデータの読み書きを実現できる。一

度、記憶場所管理テーブルを作成すれば、フラッシュROM15に対するセクタの書き込みやガベージコレクションによって記憶場所に変更が生じた場合に限って記憶場所管理テーブルの記憶位置を更新するだけで常に正しい記憶場所管理テーブルを維持することが可能である。

【0061】図10は、DRAM上に作成された記憶場所管理テーブルを説明する図である。図中、右側にRAM上の作成した記憶場所管理テーブル140を示した。0セクタと4セクタは記憶場所不在を意味する値(NULL)が入っている。これらは、フォーマット後そのセクタに対する書き込みが全く無かったか、もしくは、ファイルシステムが開放したセクタである。

【0062】ファイルシステムがファイルの消去などで不要となったセクタを解放する命令をドライバに出した場合のデバイストライバの動作は次のようになる。まず、DRAM14上の記憶場所管理テーブル140の指定されたセクタのポインタを参照してフラッシュROM15上の現在使用中の該当するセクタを探し出す。そして、当該セクタの使用済フラグをTRUEにし、DRAM14上の記憶場所管理テーブル140の指定セクタのポインタへ不在値(NULL)を代入する。

【0063】なお、ガベージコレクションの為にセクタの内容をDRAM14へ待避している場合は、記憶場所としてDRAMへのポインタが代入されている。また、同一論理セクタに対する同時操作を禁止する為のロック変数もテーブルに納めることが望ましい。

【0064】<MS-DOSのファイル復元>本電子カメラとパーソナルコンピュータ22で、記憶媒体上のデータ交換が出来ると都合が良い。本実施形態で説明したフラッシュROM管理方式を使用して、現在パーソナルコンピュータで普及しているMS-DOS(商標)と互換性があるファイルシステムを実装することができる。MS-DOSには一度消去したファイルを復元するユーティリティが付属している。ところが、本実施形態ではフラッシュROMの消去効率を向上させる為に、消去したセクタのデータ部を失ってしまう様な構成となっている。カメラで消去した媒体をパーソナルコンピュータで復元する事が原理的にできない構成になっているのである。

【0065】パーソナルコンピュータでのファイル復元機能を禁止できれば、このような事故を防ぐことができる。本実施形態では、MS-DOSがファイル復元の時に使用するデータを破壊することでファイル復元機能を禁止する。これをいかに説明する。

【0066】MS-DOS(商標)で、ファイルを消去するとディレクトリに空きスロットができる。ディレクトリにはファイル名/タイムスタンプ/最初のクラスタなどの情報が格納されている。図45はディレクトリリストの特徴を表す図である。ディレクトリリストの最後には、リストの最後であることを示すEndOfDirが格

納されている。

【0067】今、File Bを削除すると、ファイル名の先頭が削除を表すシンボルに置き換えられ、FATのクラスタチェーンが消去される。この様子を図46に示す。

【0068】アンデリートプログラムは、2番目のスロットに残った情報を元に、ファイルの復元を試みる。逆にこの情報がなければ、ファイルの復元を防止できる。

【0069】図47は本実施形態のDOS互換ファイルシステムでファイルを消去した後の状態を表している。本実施形態では、ディレクトリエンタリテーブルの最後に格納されているファイルを消去したいファイルのエンタリに上書きし、ディレクトリエンタリテーブルの最後のファイルだった部分にEndOfDirを上書きするように構成する。こうすることにより、ファイル復元機能によるファイルの復元を防止できる。

【0070】なお、ファイルの消去時にはMS-DOSと同様にセクタのデータをそのまま残しておき(セクタの開放を行わない)、ガベージコレクション時にまとめてFATとデータの関係を参照しながら不要部分を消去する方法もある。

【0071】<バックグラウンドで前処理>あるフラッシュROMでは、消去前のデータが“0”になっている方が高速に消去処理できる。フラッシュROMの消去完了の確認は、データ書き込み時と同様にデータポーリングによって行われる。従って、このようなフラッシュROMを使う場合は、バックグラウンド処理で“使用済”となったセクタのデータを0に書き換える「前処理」を行うことで性能を向上させることができる。最も低いプライオリティのタスクとして実行するようにしておけば、スループットの低下にはつながらない。

【0072】この前処理バックグラウンドでの“前処理済セクタ”管理の為にフラグを用意しておけば前処理の効率を上げることができる。

【0073】そのために、セクタのとりうる状態として、「未使用」「使用中」「使用済」に加えて「前処理済」の4つの状態を表示できる管理フラグをフラッシュROMのセクタ内部へ用意すると効率が良い。

【0074】図38は、本実施形態における消去処理速度向上のための前処理の制御手順を表すフローチャートである。同図において、ステップS2501にて、使用済でかつ前処理の済んでいないセクタを抽出する。これは、セクタ内の管理フラグが「使用済」となっていて、かつ「前処理済」となっていないセクタを抽出することで実現できる。ステップS2502において、抽出されたセクタに対してデータ「0」を上書きを開始する。ステップS2503では当該セクタについて前処理を終了したか否かを判断する。フラッシュROMへの書き込みは1バイト単位であるので、1セクタ分のバイト数の書き込みが必要となる。当該セクタに対する前処理が終了していないければステップS2504へ進み、他のタスクへ制

御を移す。

【0075】上述したように本処理は最もプライオリティの低いタスクで行われるので、CPU5がアイドル状態となったときに再び本処理が実行される。この場合処理はステップS2503へ戻る。この時点で、前回の書き込みが終了していなければそのまま他のタスクへ処理を移行する。

【0076】以上のようにして当該セクタの全バイトに対して「0」の書き込みを終えると、ステップS2503からステップS2505へ進み、当該セクタの管理フラグを、「前処理済」を示す状態にセットする。そして、引き続き、他のセクタについて前処理を行うために、ステップS2501へ戻る。

【0077】<FATキャッシュ>本システムでは、書き込み発生の度に記憶場所を変更し、その度に「未使用セクタ」が発生する。そこで、使用頻度の多い部分を特に優先的にバッファリングするキャッシュが有ればトータルの書き込み頻度が激減する事が予想される。キャッシュとして用意するメモリは多ければ多いほど良いが、システムのメモリには限界がある。

【0078】本来使用頻度の高いセクタのデータは、キャッシュ中に存在する確率も高いが、使用頻度の低いセクタを大量に読み書きした場合、当然キャッシュから吐き出されることになる。

【0079】そこで、ファイルシステムが管理する管理領域を優先的にキャッシングする様に構成すれば、スループットの向上を期待できる。なぜならファイルシステムの管理領域は頻繁に更新されているからである。

【0080】パーソナルコンピュータで普及しているMS-DOSのFATシステムの場合、720kや1.4Mといったフォーマット形式では1クラスタが1セクタで構成されている為、シーケンシャルにファイルを読む場合でも2回に1回はFATを読まなければならない。ファイルを書く場合は、さらにたくさんのFATアクセスが発生する。このため、システム中にたくさんのファイルがオープンされるとキャッシュのヒット率が落ちてしまう。

【0081】アプリケーションソフトウェアにもよるが、FATシステムにおいてFATのみを対処にしたキャッシュは、DISK全体を対象にしたキャッシュに対して1/2のメモリで同等のヒット率を確保できる。図11はキャッシュソフトウェアの階層的な位置付けを表す図である。キャッシュのソフトウェアは図11の様にファイルシステムとフラッシュROMの中間的な場所となる。

【0082】図12はキャッシュの主記憶上のデータ構造を表わす図である。片方向線形リスト構造でバッファ全体を管理している。検索方向順にデータが古くなっている。論理セクタ番号が12, 11, 6, 5の順番でアクセスすれば、図12に示されるような順番となる。ま

た、各セクタには、変更フラグが設けられており、キャッシュ上でデータの更新があった場合、変更フラグがFALSEからTRUEに変化する。このようなFATキャッシュの読み出し手順、及び書き込み手順を図13、14を参照して説明する。図13はFATキャッシュの読み出し手順を表すフローチャートである。図14はFATキャッシュの書き込み手順を表すフローチャートである。

【0083】図13において、ステップS1501でNセクタの読み出しを開始する。ステップS1502でNセクタがFATかどうかを判断する。FATでなければ、ステップS1509でフラッシュROM15からデータを読み出す。

【0084】一方、ステップS1502でNセクタがFATならステップS1503へ進み、キャッシュリストを検索する。ここでは、図12で説明した片方向線形リストを検索することになる。キャッシュ中にNセクタが存在すればステップS1507へ進み、Nセクタのバッファからデータを読み出す。

【0085】また、ステップS1503でNセクタがキャッシュリスト中に存在しなかった場合は、ステップS1504へ分岐し、最も長くアクセスされていないセクタのデータ（図12ではセクタ番号12のデータ）の吐き出しを行う。まず、ステップS1504では、キャッシュリストの最後の項の変更フラグを判断する。もし変更フラグがTRUEなら、ステップS1505へ進み、変更内容をフラッシュROM15へ書き込む。変更が無い（変更フラグがFALSEの場合）なら、そのままステップS1506へ制御を移す。読み出し手順の中で書き込みを行うのは奇妙に思うかもしれないが、バッファがキャッシュの吐き出しが起こるまで極力書き込み動作を行わない方が効率が良い。

【0086】ステップS1506でフラッシュROM15からリスト最後のバッファへNセクタの内容を読み出す。ステップS1507でNセクタのバッファからデータを読み出す。ステップS1508でNセクタのバッファをキャッシュリストの先頭へ移動させる。これは、図12において、各セクタが有する「次のバッファー」（次のバッファーを示すアドレス）の値を変更することで達成される。FATキャッシュへのアクセスが行われる度にステップS1508の動作が繰り返されることで、自然にアクセスされないバッファがリストの先頭から最後に向かってシフトしていく。よって、ステップS1504でリスト最後のバッファを選ぶのは、最も古いバッファを吐き出す為である。

【0087】次に図14を参照して書き込み手順を説明する。

【0088】ステップS1600でNセクタの書き込みを開始する。ステップS1601では、NセクタがFATかどうかを判断する。FATでなければ、ステップS

1608へ進み、フラッシュROM15へのデータの書き込みを実行する。

【0089】一方、ステップS1601でNセクタがFATならば、ステップS1602へ進み、キャッシュリストを検索する。キャッシュ中にNセクタが存在すればステップS1606へ進み、Nセクタのバッファへデータの書き込みを実行する。

【0090】また、ステップS1602でNセクタがキャッシュリスト中に存在しなかった場合は、ステップS1603へ分岐し、バッファから最も長くアクセスされていないセクタの吐き出しを行うとともに、Nセクタをキャッシュに登録する。まず、ステップS1603でキャッシュリストの最後の項の変更フラグを判断する。もし変更フラグがTRUEなら、ステップS1604で変更内容をフラッシュROMへ書き込み、ステップS1605へ進む。また、変更が無いなら（変更フラグがFALSEなら）そのままステップS1605へ制御を移す。ステップS1605では、キャッシュリストの最後の項をNセクタとする。その後、ステップS1606で、Nセクタのバッファへデータの書き込みを実行する。

【0091】その後、ステップS1607でN、セクタのバッファをキャッシュリストの先頭へ移動させる。書き込み手順の中でフラッシュROMへの書き込みを行わないのは奇妙に思うかもしれないが、キャッシュの吐き出しが起こるまで極力書き込み動作を行わない方が効率が良い。

【0092】また、ステップS1501及びステップS1601における、FATの判断であるが、ICカード等の完全に上位層（ファイルシステム）の情報を共有できないシステムでも、書き込みデータの内容を解析することでFAT領域の場所を特定できる。なぜならば、論理セクタ0に相当する部分にFATの位置等の情報が格納されていることが決まっているからである。

【0093】<フラッシュROMへの1バイトの書き込み>フラッシュROM15に対する全ての（管理領域を含む）読み書きは、最終的に1バイトの読み書き命令によって実行される。フラッシュROM15の書き込みには、通常のPROM同様の時間がかかる。1バイトの書き込みが終了するまでは、同じチップへの書き込みはできない。書き込み終了信号として信号線が用意されているチップと特別な信号が用意されていないチップがある。後者の場合は、データポーリングと言う手法で書き込み終了を確認しなければならない。データポーリングとは、ベリファイに非常によく似た方法で、書き込みデータと読み出しデータが一致するまで待つビジー制御方法である。

【0094】信号線によって書き込み終了を知ることが出来る場合は、CPU5への割り込みと併用して書き込み待ち中のCPUタイムを別のタスクへ割り当てることができます。

【0095】上述のように、信号線が無いチップの場合は、データポーリングを行なわなければならない。データ書き込みの効率をあげる為にはいくつものチップに対してパイプライン的に書き込みを行い、データポーリング時間のロスを押さえなければならない。そのため、1バイトの書き込みが完了する前に次の動作へ制御を移す必要がある。新たな読み書きを行う前に以前の書き込みが完了しているかどうかを確認するのが良い。図15は、その様子をC言語で表現したものである。

【0096】図15の1行目は、データ書き込みを行う関数の入り口である。最初の引数は最後に書き込んだアドレスとデータを保存するための構造体へのポインタ、第2の引数は書き込むアドレス、第3の引数は書き込むデータである。

【0097】3行目では、最後に書き込んだアドレスを参照してチップに書かれたデータと最後に書いたデータを比較して、両者が一致するまでループを実行する。これがデータポーリングである。前回の書き込みが完了するとこのループから抜け出す。

【0098】4行目で新しいアドレスへDataを書き込む。5行目と6行目で、今回書いたアドレスとデータを保存する。この情報は、次のデータポーリングで利用される。

【0099】リスト7行目のRotateRdyQueueは、自タスクの次に実行されるべき同一プライオリティの実行可能状態のタスクへCPUを譲るオペレーティングシステムのシステムコールである。

【0100】9行目は読み出し関数の入り口である。第1の引数はアドレスとデータを保存するための構造体へのポインタ、第2の引数は読み出すアドレスである。この関数は上位のプログラムに対して第2の引数で指定されたアドレスに格納されたデータを返す。

【0101】11行目では、もし読み出そうとしたアドレスが最後に書き込んだアドレスなら戻す値は最後に書き込んだデータなので構造体の中に保存された情報を返す。12行目は3行目と同じようなデータポーリングである。データポーリングに成功しないと同じチップの別のアドレスを読むことができない。データポーリングが終わって13行目で指定したアドレスの内容を戻している。

【0102】1チップへの書き込みを以上の様な構成にしておけば、チップ数と書き込みタスクを増やすだけで確実に見かけ上の書き込み速度を向上させることができる。また、全体のスループットを上げる為にわざとチップ数分のセクタバッファを用意（2チップなら2セクタ）して書き込む内容がバッファに溜まるまで処理しないようにすると効果がある。

【0103】図15のプログラムの特徴的なところは、データポーリングをデータ書き込み直後に実行するのではなく、次の書き込みの前に行うことである。そのために前

回書いたアドレスとデータを保存しておくRAM領域をチップごとに確保し構造体「struct DEV」として格納しているのである。

【0104】図39は、本実施形態におけるフラッシュROMへの1バイトデータの書き込み手順を表すフローチャートである。本フローチャートは、1つのフラッシュROMチップへの書き込みの制御手順を示している。ステップS2601では、前回の書き込み処理が完了したか否かを判断する。前回の書き込み処理が終了していないければステップS2604へ進み、そのまま他のタスクへ制御を移す。

【0105】一方、前回の書き込み処理が終了していれば、次の書き込みデータを準備し、これをDRAM14へ保存する。上述のステップS2601における書き込み終了の判断は、フラッシュROMに書き込まれたデータと、このステップS2602で保持されたデータとの比較によって行われる。

【0106】続いて、ステップS2603において、データの書き込みを開始する。以上のような処理によれば、複数のフラッシュROMチップに対して、複数のタスクで書き込みを行う場合に、いわゆるラウンドロビン方式を適用した書き込み処理が可能となり、複数のROMチップに対して効率良くデータの書き込みが行える。なお、マルチタスクの管理プログラムは、上述のROM13に格納されている。組み込みようのリアルタイムOSとしては、VxWorks（商標）やpSOS（商標）等が市販されており、ROM13にこれらのようなリアルタイムOSが格納されている。

【0107】<フラッシュROM書き込み電源の共有化>データの書き込みや消去の際にPROM同様に12V等の特別な書き込み電圧を必要とするチップや、書き込み電圧を与えることで書き込みが高速になるチップがある。この様なチップを使用する場合に専用のDC/DCコンバータ等の電圧発生部を設けると電子カメラのコストアップにつながる。ところが、従来よりカメラにはストロボの充電や、機構部分やCCDの駆動等、特別な電圧が必要な部分がありDC/DCコンバータ等を搭載している。そこで、フラッシュROMの書き込み電圧とストロボ充電やメカ駆動を時分割多重で行うことで、少容量のDC/DCコンバータでシステムを構築でき、システムのコストを押さえることができる。

【0108】図16は、DC/DCコンバータの出力容量を越えない様に電源を管理するプログラムをC言語で表現したものである。Line1~6が1ステップのズームアップ関数で、Line7~13がフラッシュROMへ1セクタ書き込む書き込み関数である。ズームアップ関数はLine3でDC/DCコンバータの資源管理用のセマフォ“SemDCDC”を獲得して、モータを1ステップ動かす関数をLine4で呼び出す。モータ駆動が終わるとDC/DCコンバータの資源管理用のセ

マフォ“SemDCDC”を開放する。セマフォはマルチタスクのオペレーティングシステムで資源を管理する為の一般的な方法であり、多くのオペレーティングシステムがシステムコールとして用意している。

【0109】即ち、Line3で既に“SemDCDC”が他のタスクによって使用されていたとすると、他のタスクがセマフォ“SemDCDC”を開放するまでズームアップをしようとしたタスクの実行が保留される。

【0110】書き込み関数はLine9でセマフォ“SemDCDC”を獲得し、フラッシュROMへ1セクタのデータを書き込む。Line11でデータポーリングを行い最後の書き込みが終了したことを確認したら、Line12でセマフォ“SemDCDC”を開放する。このようにプログラムを構成すれば、ズームアップとフラッシュROMの書き込みを同時に行なうことは無くなる。ズームは1ステップ単位であり、書き込みはセクタ単位なので非常に短い保留時間の後に必ず電源を獲得できる。

【0111】図16について更に説明すると、図16のLine1はズームアップする関数の入り口である。本関数には引数はない。Line3で電源の使用権利として宣言したSemDCDCの権利を一つ獲得する。この時、使用権利が1つもなければこの関数を呼び出したタスクの実行は保留される。電源の使用権利を別のタスクが解放すればZoomUpを呼び出したタスクが再び実行可能状態に戻る。そして、Line4のモーターを動かす関数を呼び出すことができる。そして、Line5で、電源使用権利を返却してこの関数の仕事は終了する。Line7は1セクタのデータをEEPROMに書き込む関数の入り口であり、Line9で電源利用権利を獲得してLine12で返却している。

【0112】図40は、上述した電源の共有手順を説明するためのフローチャートである。同図において、ステップS1701で、電源コントローラ9によるDC/DCコンバータ8の出力電力の供給が解放されたか否かを判断する。ステップS1702では、電源確保のための指示の内容を解析し、この指示結果に従って、ステップS1703、1705、1707、1709のいずれかに分岐する。

【0113】指示の内容が、CCD駆動電力の供給であれば、ステップS1703へ進み、CCD2に対してCCD駆動のための電力を供給する。そして、ステップS1704にて、CCD駆動の終了（即ち撮影動作の終了）を検出すると、ステップS1711へ進み、電源の解放を行う。また、ストロボの充電要求であれば、ステップS1705へ進み、電源コントローラ9に対してストロボ21に対する充電電力を提供させる。そして、ステップS1706でストロボの充電を完了したら、ステップS1711へ進み、電源の解放を行う。なお、充電

の電力供給は、所定時間の充電を行う毎に他の電源供給のために電源を解放する。即ち、ストロボ21への充電を管理するプログラムは別個に所定のタスクに存在し、充電の完了はそのタスクによって管理される。

【0114】指示の内容が、ズーム機構の駆動であれば、ステップS1707へ進み、ズーム機構の駆動系(不図示)へ電力供給を行う。そしてステップS1708で、1ステップのズーム動作を終えたらステップS1711へ進み、電源を解放する。更に、指示の内容がフラッシュROMへの書き込みであれば、ステップS1709へ進み、フラッシュROM15への書き込み電力を供給する。1セクタ分の書き込みが終えたら、ステップS1710からステップS1711へ進み、電源を解放する。

【0115】なお、ステップS1704、1706、1708、1710において、各動作の終了を待つが、この待ちループにおいて、他のタスクへの制御が移り、マルチタスク処理が遂行される。この管理処理は、各タスクから随時起動が可能であり、複数のタスクで同時に起動される可能性もある為、ステップS1701で電源解放のチェックを行っている。

【0116】以上の図40のフローチャートによれば時分割で電源を利用することが可能となる。しかしながら、すべてのシステム(CCD/ストロボ/ズーム/フラッシュROM)が依存しあった1つのプログラムである。このようなソフトウェアを開発すると、開発/デバッグ/メンテナンスのコストが大きくなり、拡張性や柔軟性を保つのが難しくなる。

【0117】そこで、電源を1つの資源に見立ててOSの提供する資源管理機能を用いることで開発効率を向上させることができる。そこで上述のセマフォによる資源管理を行う。即ち、CCDの駆動部、ストロボの駆動部、ズームの駆動部、フラッシュROMの駆動部のそれぞれの制御プログラムが、電源という資源(セマフォ)を獲得、解放することで、時分割された電源の割当てが行える。

【0118】図48は本実施形態による電源の時分割利用を説明する図である。同図に示されるように、あるタスクA(例えばCCD)によって電源要求が発生したとき、電源セマフォが解放された状態にあれば、そのセマフォを獲得して、電源を占有する(ステップS2001～S2003)。続いてステップS2004において、当該電源よりの電力供給を得て所定の処理を行うと、ステップS2005へ進んでセマフォを解放する。

【0119】一方タスクAより遅れて電源獲得を要求したタスクBでは、ステップS2011における電源要求ではセマフォを獲得できず、ステップS2012により、セマフォの解放待ちとなる。そして、タスクAよりセマフォが解放されると、このセマフォをタスクBが獲得して、電源を占有する(ステップS2013)。その後タスクBで所定の処理を実行し(ステップS201

4)、電源を解放する(ステップS2015))。

【0120】以上のようなセマフォによる電源資源の管理により、電源の時分割利用が可能となる。

【0121】なお、図48によれば、電源資源の利用権利を示すセマフォが一つしかないが、複数個のセマフォが存在するようにも良いことは言うまでもない。

【0122】<実施形態の電子カメラの動作説明>図17は、本実施形態のリブートからサービスの開始までの動作手順を表わすフローチャートである。ステップS101でシステムがリブートすると、ステップS102でフラッシュROM15の管理領域をスキャンし、DRAM14上に記憶場所管理テーブル140を作成する。また、この処理と並行して、DRAM14上の未使用セクタカウンタ、使用済セクタカウンタ、使用中セクタカウンタへ、それぞれの状態に対応するセクタがいくつ有るかを数え、セットする。このカウンタは、後にフラッシュROM15に対して操作を行ったときに更新され、記憶効率を判断するのに用いられる。その後、ステップS103へ進み、各種のサービスを開始する。

【0123】図18は、指定セクタの読み出しサービスの手順を表わすフローチャートである。まず、ステップS201でNセクタの読み出しを開始する。ステップS202では、Nセクタをロックする。セクタのロックはロック変数を使って行う。このロック変数は、記憶場所管理テーブル140で各セクタの記憶場所とともに管理される。ステップS202では、セクタが既に他のタスクによってロックされている場合、他のタスクによってアンロックされるのを待ち、他のタスクによってアンロックされた後で当該セクタのロックを行う。ロックしたセクタはステップS206でアンロックするまでの間、自タスクによって占有することが出来る。

【0124】ステップS202で論理セクタをロックすると、ステップS203で記憶場所管理テーブルを参照して、当該セクタに有効なデータ記憶されているかどうかを確認する。有効なデータが記録されて無い場合は、ステップS204へ分岐する。ステップS204では、ダミーのデータ(例えば全部0など)をセクタの内容として読み出す。ステップS203で有効なデータが格納されていると判断された場合は、ステップS205へ分岐する。ステップS205では記録場所管理テーブルの値を元にフラッシュROM(または主記憶)からデータを読み出す。

【0125】ここで、ガベージコレクションを実行中でNセクタが主記憶(DRAM14)へ退避されていた場合は、記憶場所管理テーブルのポインタは主記憶をポインティングとしている。また、図中点線で囲んだ部分はNセクタを占有している期間である。この様なロック機構によって1つのセクタ操作の安全性を保証している為、ガベージコレクションの途中でも操作中でないセクタを自由に読み出すことが可能となっている。

【0126】図19は論理セクタの書き込みサービスの手順を表わすフローチャートである。ステップS301でNセクタの書き込みを開始する。ステップS302でステップS202と同様に、論理セクタのロックを行う。

【0127】次に、ステップS303で、記憶場所管理テーブルを検索して、Nセクタに有効なデータが記録されているかどうかを判断する。有効なデータが記録されていればステップS304へ、記録されていないならばステップS305へそれぞれ分岐する。ステップS304では、それまで有効なデータとして記録されていたフラッシュROM（または主記憶）のデータを破棄する。ステップS304におけるデータ破棄の処理は、図21のフローチャートを用いて詳しく説明を加える。ステップS304の後ステップS305へ制御が移る。

【0128】ステップS305では、フラッシュROM15においてNセクタを書き込むための記憶領域を獲得する。ステップS305における記憶領域の獲得手順は図23を用いて詳しく説明を加える。ステップS305で正常に記憶領域の獲得に成功すれば、ステップS308へ制御を移す。ステップS308では獲得したフラッシュROM15の領域へNセクタのデータを書き込む。

【0129】一方、ステップS305でフラッシュROMに記憶場所が無い場合、即ち記憶領域の獲得に失敗した場合はステップS306へ分岐する。ステップS306はデータの退避用に主記憶を獲得する。主記憶の領域確保はオペレーティングシステムが提供するメモリ管理機能によって行う。これはC言語でalloc関数に相当する機能である。そして確保した領域を片方向線形リスト構造によって管理する。

【0130】図20は主記憶上に獲得した退避データリストの様子である。（a）は退避データリストにデータが無い状態であり、リストには、END_OF_LISTが代入されている。（b）は退避データリストに、セクタ番号3, 20, 221の各セクタの内容が退避されている状態である。

【0131】ステップS309で、記憶場所管理テーブルを更新する。ここで、記録したフラッシュROM（または主記憶）へのポインタが代入される。ステップS310で論理セクタのアンロックを行う。図中点線で囲まれた期間その論理セクタを占有できる。ステップS311で記憶効率の評価を行う。記憶効率の評価手順については、図21のフローチャートを用いて詳しく説明を加える。記憶効率の評価の結果、記憶効率が悪化した場合は、ステップS312へ制御を移す。ステップS312では上述したガベージコレクションを行う。ガベージコレクションについては、図24のフローチャートを参照して詳しく説明を加える。ステップS313でNセクタの書き込みが終了してメインのルーチンへ復帰する。

【0132】なお、記憶場所管理テーブルに納められる

のは、記憶場所のポインタ（バス空間上のアドレス）である。図20の（b）の主記憶に待避されたデータの

「次のデータへのポインタ」の次のフィールド（図中ではすぐ下に示されている）からは、図10の左側にあるフラッシュROM上のデータ構造と互換性がある。記憶場所管理テーブルに納められるのはこの互換部分へのポインタである。このように構成することにより、データの読み出しプログラム側でフラッシュROMと主記憶を単一のアルゴリズムで扱うことが可能となる。

【0133】次に、指定されたセクタの記憶を破棄する手順（上述のステップS304）を説明する。図21は、記憶を破棄する手順を表わすフローチャートである。

【0134】ステップS401で指定領域の記憶破棄を開始する。ステップS402では、指定されたセクタを記憶する領域が主記憶上にあるかどうかを判断する。主記憶上にあるならステップS405へ分岐する。ステップS405で待避セクタリスト（本例では、図20で示した片方向線形リスト）から指定領域を削除する。

【0135】片方向線形リストからの指定領域の削除手順は、まずリストの先頭から検索方向順にリストをたどり、ポインタが自分をポインティングしている項を検出する。そして、この検出された項のポインタに現在自分がポインティングとしている値を代入することで実現する。そして、ステップS406で、リストから削除した主記憶領域をオペレーティングシステムへ返却する。オペレーティングシステムへの記憶領域の返却はC言語のfree関数に相当する機能である。

【0136】一方、ステップS402で指定された領域が主記憶上ではない（すなわちフラッシュROM上）ならステップS403へ分岐する。ステップS403では、指定されたフラッシュROM上のセクタの管理フラグを“使用済”へ変更する。これは、使用済みフラグをTRUEにセットすることで達成される。ステップS404では主記憶上の未使用セクタカウンタの値を1つ減少させる。ステップS407で復帰する。

【0137】次に、記憶効率の評価手順（ステップS311）について説明する。図22は、記憶効率の評価手順を表わすフローチャートである。

【0138】ステップS501で記憶効率の評価を開始する。ステップS502では、主記憶に設定された未使用セクタカウンタの値と使用済セクタカウンタの値を比較する。ここで、使用済セクタカウンタの値が未使用セクタカウンタの値に対して同じか上回った場合、上位プログラムに対して記憶効率の悪化をレポートする様に構成している（ステップS502、S504）。また、未使用セクタカウンタの値が使用済セクタカウンタの値よりも大きければ、評価結果を正常とし、正常復帰する（ステップS503）。

【0139】次に、フラッシュROMの記憶領域の獲得

手順（ステップS305）について説明する。図23はフラッシュROMの記憶領域の獲得手順を表わすフローチャートである。

【0140】ステップS601でフラッシュROMの記憶領域の獲得を開始する。ステップS602で未使用セクタの検索権利を獲得する。ここでは、オペレーティングシステムの提供するセマフォの機能を使用して未使用セクタの検索権利を管理している。ここでは、ステップS602からステップS609／ステップS611までの点線で囲まれた処理期間だけ未使用セクタの検索権利を独占出来る。複数のタスクが同時に同一領域を獲得する様な事態を防ぐ為のしくみである。

【0141】ステップS603でフラッシュROMの最初のセクタへポインタを移動する。ステップS603でそのセクタの管理フラグ（使用中フラグ、使用済みフラグ）を参照して、当該セクタの使用状態を判断する。使用済みか使用中ならステップS605へ分岐する。ステップS605で現在ポイントしているセクタが最後のセクタならステップS611へ分岐する。この場合、使用可能な領域がフラッシュROM15に存在しないことになるので、ステップS611で未使用セクタの検索権利を開放した後、ステップS612で異常復帰する。また、ステップS605で現在ポイントしているセクタが最後のセクタでなければ、ステップS606へ分岐する。ステップS606では、ポインタを次のセクタへ移動させてからステップS604へ戻る。

【0142】ステップS604ポインタの示すセクタの管理フラグが未使用となっていればステップS607へ分岐する。ステップS607では、フラッシュROMの管理フラグを“使用中”へ変更する（使用中フラグをTRUEにする）。そして、ステップS608で、主記憶に設けた未使用セクタカウンタの値を1つ減少させる。この場合は、フラッシュROMへの記憶領域の獲得に成功しているので、ステップS609で未使用セクタの検索権利を開放し、ステップS610で正常復帰する。

【0143】次に、ガベージコレクション（ステップS312）の手順について説明する。図24はガベージコレクションの手順を表わすフローチャートである。

【0144】ステップS701でガベージコレクションを開始する。ステップS702では、整理対象のイレースブロック（以後、整理対象ブロック）を選出する。整理対象ブロックの選出手順については、図25のフローチャートを用いて詳しく説明を加える。ステップS703では、整理対象ブロックの未使用セクタを使用済化する。この使用済化の手順については、図26のフローチャートを用いて詳しく説明を加える。ここで、最初に整理対象ブロック内の未使用セクタを使用済化させる目的は、ガベージコレクション中であっても、他のタスクが整理対象ブロック内のセクタを含むセクタへの読み書きが可能な構成となっており、ガベージコレクション中に

他のタスクによって整理対象ブロック内のセクタへ新たなデータが書き込まれることを防止する為である。

【0145】ステップS704では、整理対象ブロック中の使用中セクタを他の記憶領域（即ち、他のイレースブロック）へ移動させる。使用中セクタを他の記憶領域へ移動させる処理については、図27のフローチャートを参照して詳しく説明を加える。

【0146】続くステップS705では、使用中セクタの移動を終了した整理対象ブロックの消去を実行する。整理対象ブロックを消去する手順については、図28のフローチャートを参照して詳しく説明を加える。なお、この整理対象ブロックの消去において、消去回数カウンタ152の内容を主記憶にコピーしておく。ステップS705における整理対象ブロックの消去を終えると、ステップS706で主記憶に退避したデータをフラッシュROMの当該イレースブロックの消去回数カウンタへ戻す。そして、ステップS707でガベージコレクションから復帰する。

【0147】次に、ガベージコレクションにおける整理対象ブロックの選出手順（ステップS702）について説明する。図25は整理対象ブロック選出する手順を表わすフローチャートである。

【0148】まず、ステップS801で整理対象ブロックの選出を開始する。ステップS802で評価ポインタに最初のイレースブロックをセットする。同様に、ステップS803で、整理対象候補ポインタを最初のイレースブロックにセットする。

【0149】次に、ステップS804で、評価ポインタの示すイレースブロックに使用済セクタが含まれているかどうかを判断する。使用済セクタが含まれていなければステップS804、ステップS805をスキップしてステップS807へ制御を移す。

【0150】一方、ステップS804で評価ポインタの示すイレースブロックに使用済セクタが含まれている場合には、ステップS805へ制御を移す。ステップS805では、整理対象候補ポインタの示すイレースブロックの消去回数カウンタの値と評価ポインタの示すイレースブロックの消去回数カウンタの値を比較する。もし評価ポインタの示すイレースブロックの消去回数の方が少なければステップS806へ制御を移す。ステップS806では、整理対象候補ポインタへ評価ポインタを代入する。一方、ステップS805でもし評価ポインタの示すイレースブロックの消去回数のほうが多ければそのままステップS807へ制御を移す。

【0151】ステップS807で評価ポインタが最後のイレースブロックを示しているかどうかを判断する。もし最後のイレースブロックでなければ、ステップS808で評価ポインタを次のイレースブロックへ移動させた後、ステップS804へ戻る。以上のように、ステップS804～S808の処理を繰り返すことで、整理対象

候補ポインタは、使用済みセクタを含み、消去回数の少ないイーストブロックを示すようになる。

【0152】ステップS807で評価ポインタが最後のイーストブロックを示している場合はステップS809へ分岐する。ステップS809ではガベージコレクション処理(図24の処理)に復帰する。この時点の整理対象候補ポインタの示すイーストブロックが整理対象として選出される。

【0153】次に、選択された整理対象ブロック内の未使用セクタを使用済み化する処理(ステップS703)について説明する。図26は、整理対象ブロックの未使用セクタを使用済み化する手順を表わすフローチャートである。

【0154】ステップS901で処理を開始する。ステップS902で、整理対象ブロックの最初のセクタへポインタを移動させる。次に、ステップS903で、未使用セクタの検索権利を獲得する。これは、図23のフローチャートのステップS602と同様の効果があり、ステップS908までの点線で囲まれた間、未使用セクタの検索権利を独占する。即ち、整理対象ブロックの全セクタを対象にスキャンして未使用セクタを使用済セクタへ変更するまでの間、他のタスクが未使用セクタの検索をすることを禁止する。しかし、管理フラグのみの操作で未使用セクタを使用済セクタへ変更するので、検索権利の独占時間は短く、全体のスループットが低下することはない。

【0155】ステップS904で、現在のポインタが示すセクタが未使用セクタかどうかを判断する。もし未使用セクタならステップS905へ分岐する。ステップS905でその記憶を廃棄する。ステップS905の処理手順は図21のフローチャートで説明した通りである。この処理により、未使用セクタが使用済みセクタに変更される。ステップS906では、ポインタが整理対象ブロックの最後のセクタを示しているかどうかを判断する。最後のセクタを示していればステップS908へ、そうでないならステップS907へ分岐する。ステップS907ではポインタを次のセクタへ移動させてステップS904へ制御を戻す。

【0156】また、ステップS906でポインタが整理対象ブロック最後のセクタならステップS908で未使用セクタの検索権利を開放し、ステップS909でガベージコレクション処理(図24のフローチャート)へ復帰する。

【0157】次に、整理対象ブロックの使用中セクタを他のイーストブロックの未使用セクタへ移動する処理

(ステップS704)について説明する。図27は、整理対象ブロックの使用中セクタの移動手順を表わすフローチャートである。

【0158】ステップS1000で処理を開始する。ステップS1001で整理対象ブロックの最初のセクタへ

ポインタを移動させる。以下のステップS1002～S1012では、ポインタが指し示すセクタについて処理を行う。

【0159】ステップS1002で当該セクタの管理フラグ(使用中フラグ、使用済みフラグ)を判断する。ステップS1002で管理フラグの値が「使用中」となっていたらステップS1003へ制御を移し、「使用済」となっていたらステップS1012へ制御を移す。ステップS1003で、論理セクタをロックする。ロックしたセクタはステップS1011でアンロックされるまでの間、自タスクで占有される。

【0160】ステップS1004では記憶領域を獲得する。ステップS1004における記憶領域の確保の手順は、図23のフローチャートで説明した通りである。ここで、整理対象ブロック内の各セクタは上記ステップS703の処理で、全て使用済み化されているので、確保される記憶領域は整理対象ブロック以外のイーストブロックとなる。

【0161】記憶領域の獲得に成功すると、処理はステップS1008へ進む。ステップS1008では、獲得した領域へ当該セクタのデータをコピーする。そして、セクタの移動に従って、ステップS1009で記憶場所管理テーブル140を更新する。

【0162】一方、ステップS1004で記憶領域の獲得に失敗した場合は、ステップS1005へ分岐する。ステップS1005では、データ退避用の記憶領域を主記憶(DRAM)より獲得する。データ退避用記憶領域の獲得は図19のフローチャートのステップS306で説明した通りである。ステップS1006では、獲得した領域へ当該セクタのデータをコピーする。そして、ステップS1007で記憶管理テーブルを更新する。ステップS1010で元の記憶を廃棄する。即ち、ポインタの指し示すセクタの使用済みフラグをTRUEにセットする。そして、ステップS1011で当該論理セクタをアンロックする。

【0163】ステップS1012で、ポインタの指し示すセクタが、整理対象ブロックの最後のセクタかどうかを判断する。最後のセクタであればステップS1014へ、最後のセクタでなければステップS1013へそれぞれ分岐する。ステップS1013では、ポインタを次のセクタへ移動させて、ステップS1002へ戻り、次のセクタについて上述の処理を繰り返す。また、ステップS1014では、整理対象ブロック内の全てのセクタについて処理を終えているので、ガベージコレクション処理(図24のフローチャート)へ復帰する。

【0164】次に、整理対象ブロックの消去処理(ステップS705)について説明する。図28は、整理対象となったイーストブロックの消去手順を表わすフローチャートである。

【0165】ステップS1101で処理を開始する。ス

ステップS1102で整理対象ブロックの消去回数カウンタを主記憶へコピーする。ステップS1103では整理対象ブロックの消去を実行する。ステップS1104では、主記憶へコピーした消去回数カウンタの値を1増加させた値をフラッシュROMへ書き込む。即ち、当該整理対象ブロックの消去カウンタの値を、消去処理前の値より1増加させる。その後、ステップS1105でガベージコレクション処理(図24のフローチャート)へ復帰する。

【0166】上記図24で示されるガベージコレクション処理は、極力フラッシュROMを用いた処理であり、退避データの安全性が高い。しかしながら、上述の<未使用セクタが無い場合>の項で説明したように、積極的に主記憶(DRAM14)を用いて使用中セクタのデータを待避し、複数個のイレースブロックを消去すると消去処理の効率がよい。但し、DRAM14にデータを待避するので、待避中のデータに関して安全性が低下する(例えば電池が外れて電源供給が停止するとDRAMに待避したデータが失われることになる)。そこで、電源の種別を判断し、供給電源が電池の場合は待避データの安全性を重視し、ACアダプタの場合は電源供給が停止する危険性が少ないので消去処理の効率を重視するように構成してもよい。この場合の処理について図36を参照して説明する。

【0167】図36は、電源種別に基づいてガベージコレクション処理を切り換える場合の処理手順を説明するフローチャートである。同図において、図24のフローチャートで示される処理と同じ処理を行なうステップについては同一のステップ番号を付し、ここでは詳細な説明を省略する。

【0168】ステップS1300においてガベージコレクション処理が起動されると、ステップS1301へ進み、当該装置への電源供給の形態を判断する。ここでは、図1の電源コントローラ9が、電源の供給元が電池7であるかACアダプタ23であるかを判断し、CPU5に通知する。電源種別が電池7であった場合は、ステップS1304へ進み、上述の図24で示したガベージコレクション処理を実行する。

【0169】一方、ステップS1301において電源種別がACアダプタであった場合は、ステップS1302へ進む。ステップS1302では、図24のステップS702、S703、S704に相当する処理を実行し、選出した整理対象ブロック内の未使用セクタの使用済み化と使用中セクタの待避を行う。そして、ステップS1303において、主記憶(DRAM14)にセクタの待避を行うのに十分な空き領域があるか否かを判断し、十分な空き領域があればステップS1302へ戻る。ステップS1302では、前回の整理対象ブロックとは別の整理対象ブロックを選出して、上述の処理を繰り返す。

【0170】DRAM14上に十分な空き領域が無くな

ると、ステップS1303からステップS1304へ進み、上述の処理で選出された整理対象ブロックの消去を行う。そして、ステップS706で主記憶に待避したデータをフラッシュROM15に戻して本処理を終了する。

【0171】以上のように、図36の処理によれば、電源がACアダプタによって供給される場合は、主記憶の空き容量を積極的に利用してデータの待避を行い、複数の整理対象ブロックを選出して、一括して消去処理を行うことができ、消去処理の効率が向上する。

【0172】なお、上記の処理では、電源種別に基づいて自動的にガベージコレクションの形態を切り換えるが、コントロールパネル12の操作により、マニュアルで切り換えるようにすることもできることはいうまでもない。

【0173】次に、基本サービスの一つである論理セクタの解放手順について説明する。図29は、論理セクタの解放手順を表すフローチャートである。

【0174】ステップS1201でNセクタの解放を開始する。ステップS1202でNセクタをロックする。この結果、ステップS1205でアンロックされるまでの間、自タスクで論理セクタを占有出来る。続いて、ステップS1203で当該セクタの記憶を廃棄する。この記憶の廃棄処理については、図21のフローチャートで説明した通りである。ステップS1204では、DRAM14の記憶場所管理テーブル140へ“不在”値を代入する。ステップS1205では、論理セクタをアンロックし、ステップS1206で復帰する。

【0175】例えばMS-DOS(商標)等の一般的なファイルシステムでは、ファイルの消去に際しては、当該ファイルに属するセクタをFATにおいて上書き可能とするのみで、各セクタを解放するということは行われない。よって、このようなファイルシステムに本実施形態のフラッシュROM管理システムを適用すると、ファイルシステム上では無効となったデータが、有効なセクタとして残されてしまうことになり、ガベージコレクション等の効率を低下させることになる。よって、ファイルシステムの指示(例えばファイル消去)に基づいて不要となったセクタを検出し、これを解放するように構成すれば、ガベージコレクションの効率をより向上させることができる。

【0176】図37は、ファイルシステムよりファイル消去が指示された場合の、不要セクタの解放手順を表すフローチャートである。同図において、ステップS1401でファイルシステムよりファイル消去の指示があったか否かを判断する。ファイル消去の指示があった場合は、ステップS1402へ進み、消去すべく指示されたファイルに含まれるセクタを抽出する。セクタの抽出は、FATを参照することで抽出できる。そして、ステップS1403で、先のステップS1402で抽出され

た各セクタについて、上記図29のフローチャートで説明したセクタの解放処理を実行する。

【0177】[実施形態2] 次に実施形態2について説明する。

【0178】<ディスクコントローラエミュレーション>上述の実施形態1で説明したフラッシュROMの記憶管理システムは、上位層から見た特徴がディスク媒体と良く似ている。従って、ディスクコントローラのエミュレーション機能を備えたシステムに組み込むことで、ディスクコントローラとディスク媒体をディスクコントローラエミュレーションと本実施形態の記憶管理システム（あるいは本実施形態の記憶管理システムを組み込んだICカード）へ置き換えることが可能となる。近年PCMCIAに代表されるICカードが普及しているが、ICカードへディスクコントローラエミュレーション機能と上記実施形態1の記憶管理システムを組むことにより、リムーバルな記憶媒体として利用することが可能となる。第2の実施形態では、実施形態1の記憶管理システムをICカードへ組み込んだものについて説明する。

【0179】図3.0は実施形態2におけるICカードの構成を表すブロック図である。同図において、200はICカード全体を示す。201はマイクロコンピュータであり、ディスクコントローラエミュレーション及び記憶管理を行う。202はROMであり、マイクロコンピュータ201のプログラムを格納する。203はRAMであり、マイクロコンピュータ201の主記憶として機能する。204はフラッシュROMであり、上記実施形態1で説明した記憶管理システムによってデータを蓄積する。即ち、フラッシュROM204は、図4で説明した管理領域とデータ領域とで管理される。

【0180】205はコマンド/データ・ラッ奇部であり、ホスト装置より受信した外部バスからのコマンドシリンド番号等を保持する。206はFIFOメモリであり、先入れ先出し方式でデータの入出力を行う。207はタブルROMであり、当該カードの特徴等を記憶しており、外部バスからのみ読み出しができる。

【0181】上述の各構成の機能は、以降の動作説明でより明らかとなる。

【0182】図3.1は、本実施形態2のICカードを利用する為のホストシステムの簡単なブロック図である。同図において、301はホストシステム側のマイクロコンピュータである。302はカードインターフェースであり、ホストシステムの内部バスとICカード200の外部バスを接続する。なお、カードインターフェース302は、ICカード200への電源供給を行うための電源供給線や、ICカード200からの割り込み要求（IRQ出力）を受け付けるための信号線も備えている。

【0183】図3.2は、図3.1のホストシステムがICカードを接続する際の手順を示すフローチャートである。ステップS4100で処理を開始すると、ステップ

S4101でICカードへの電源供給を開始する。ステップS4102では、ICカード200内のタブルROM7から、タブル形式で格納されているデータを解析する。タブルROM7の内容を解析することで、接続されているICカードの特徴が分かる。

【0184】ステップS4103では、ステップS4102で解析したタブル情報によって、接続されているICカードが内部バスへ接続可能かどうかを判断する。そして、接続可能ならステップS4104へ、接続不可能ならステップS4105へとそれぞれ分岐する。ステップS4104では、ICカード側のバスをホストの内蔵バスのメモリ空間とIO空間へマッピングする。この時点でホスト装置のバスの空間にディスクコントローラが有るのと同じ状態になる。

【0185】図3.3はICカード200内のマイクロコンピュータ1のメインシーケンスを示すフローチャートである。ステップS4201でICカードの電源が投入されると、ステップS4202で記憶管理システムの初期化を行う。即ち、フラッシュROM204の全イレースブロックの論理セクタの状態を一旦読み出し、読み出した情報に従って主記憶用のRAM203へ記憶場所管理テーブルを作成する。ステップS4203で主記憶上のコマンドバッファとしてリング状のバッファを用意して初期化し、割り込み処理を許可する。この処理以降割り込みルーチンの動作が始まる。

【0186】割り込みルーチンのシーケンスを図3.4のフローチャートに示す。割り込みルーチンの動作を理解した方が、図3.3のフローチャートの説明が容易となる為、ここで図3.4のフローチャートについて説明を行う。

【0187】ホストシステムがコマンド/データ・ラッチ205へのコマンドのアドレスへコマンドを書き込むと、コマンド/データ・ラッチ205からマイクロコンピュータ201へ割り込みが発生する。コマンド/データ・ラッチ205は、ホストバスとICカード内部のバスのIOアドレス空間にマッピングされていて、コマンド/データはそれぞれ図3.5に示すようにIOアドレスが割り振られている。図3.5は、本実施形態のコマンド/データ・ラッチにおけるIO割り付けを示す図である。本例では、図3.5中のCommandのアドレスにコマンド（例えばデータの読み出しを指示するReadSector(s)）を書き込むことでマイクロコンピュータ201へ割り込みが発生する。

【0188】割り込みが発生すると、マイクロコンピュータ201のソフトウェアは、図3.4のフローチャートのステップS4301へ制御を移す。ステップS4302では、コマンド/データ・ラッチ205に書き込まれたデータを読み出して、主記憶上のリングバッファへデータを格納する。ステップS4303で割り込みルーチンを終了して図3.3のフローチャートへ復帰する。

【0189】図33のフローチャートの説明に戻る。ステップS4204でマイクロコンピュータ201はコマンドバッファの状態を判断する。コマンドバッファヘデータが格納されていれば、ステップS4205へ分岐し、データが格納されていなければステップS4213へ分岐する。ステップS4213ではCPUを休止状態にする。多くのワンチップマイクロコンピュータは、命令の実行を休止して消費電流を減らす機能を備えているが、本実施形態のCPUもこの種の機能を備える。そして、IRQによる割り込み要求信号が入力されると、CPU201は休止状態から復帰して上述の割り込みルーチンを実行する。割り込みプログラムの実行が済んだ時点でステップS4213から復帰してステップS4204へ戻る。

【0190】ステップS4204でコマンドバッファヘデータが格納されていると、ステップS4205へ移行する。ステップS4206では、リングバッファからデータを読み出す。ステップS4206でコマンドを解釈する。Seekコマンドの場合はステップS4207、ReadSector(s)コマンドの場合はステップS4208へ、WriteSector(s)の場合はステップS4209へ、IdentifyDrvコマンドの場合はステップS4210へそれぞれ分岐する。他にもコマンドがあるが本実施形態の説明上重要なものは省き、フローチャートを簡略化している。ステップS4207～4210までのコマンドの実行を終了したらステップS4204まで戻り、上記の処理を繰り返す。

【0191】ステップS4207では、Seekコマンドを実行する。SeekといつてもフラッシュROMには、ディスクデバイスと違ってヘッドが無いので、次のコマンドに備えての妥当性等をチェックするだけである。ICカードのサポートするヘッド数を超えるヘッド位置などを指定された場合は、ディスク装置同様にエラーが発生する。

【0192】ステップS4208はReadSector(s)コマンドに対する処理を行う。ReadSector(s)コマンドは、読み出すべきセクターの個数が図35のSectorCountで指定される。よって、ステップS4208では、指定された場所のセクタをSectorCount個読み出す行為を行う。本実施形態の記憶管理システムでは、リニアな論理セクタ番号を使って管理を行っているので、シリンド／ヘッド／セクタ番号を元にリニアな論理セクタ番号を計算し、論理セクタの内容をFIFOメモリ206へ転送し、コマンド／データ・ラッチ205のSectorNumberのインクリメントも行う。FIFOメモリ206は、ICカード200の内部バスから書き込んだデータを外部バスから読み出すことができ、また外部バスから書き込んだデータをICカード内部バスから読み出す構成となつたFIFOメモリである。

【0193】ここで、上述のリニアな論理セクタ番号に

ついて説明する。一般にハードディスクに対して指定する番号は、セクタ、シリンド、ヘッドのパラメータで決まる3次元の不連続な番号である。例えば、シリンド数が1024個、ヘッド数が16個、セクタ数が63個のハードディスクの場合、セクタ数は $1024 \times 16 \times 63 = 1032192$ 個となる。

【0194】このセクタを0番から1032192番としてアクセスできると良いのであるが、上記の3つのパラメータをすべて指定してアクセスするように設計されている。例えば、シリンド500・ヘッド16・セクタ63の次は、シリンド501・ヘッド0・セクタ1をアクセスするといった具合である。なお、これら3つのパラメータをそれぞれの頭文字をとってCHSパラメータと呼ぶ。

【0195】MS-DOS(商標)のようなオペレーティングシステムでは、内部ではリニア(連続的)なセクタ番号を用いるが、デバイスドライバがこれをCHSパラメータに変換する。本実施形態のシステムでは、リニアなセクタ番号を用いるのでCHSパラメータの値を元にリニアなセクタ番号を求める。上記で挙げたハードディスクの場合は、

シリンド番号×(16×63)+ヘッド番号×(63)
+セクタ番号

を計算することで、リニアな論理セクタ番号が求まる。

【0196】ステップS4209はデータラッチで指定された場所のセクタへデータを書き込む処理を行う。データは、FIFOメモリ206経由でホストシステムから受け取る。

【0197】ステップS4210は、ICカード200がどのようなハードディスクをエミュレーションしているかという情報を返す処理を行う。すなわちシリンド数やModelNumberなどハードディスクとしてのスペックを含むデータをFIFOメモリ206へ書き込む処理を行う。

【0198】<ファイルシステムの解析>以上説明した様に実施形態1で説明した記憶管理システムをICカードに組み込むことで、ATAハードディスク等の置き換え用途に使用できる。しかし、ATAコマンド等のFATキャッシュやファイル消去によって生じた不要セクタの開放といった処理を行う為の情報を上位システムからもらう手法が無い。ATAコマンドの空き部分を利用してセクタの開放コマンドとキャッシュするセクタ番号指定コマンドを追加実装することで、FATキャッシュと不要セクタ解放の機能が実現できる。そして、この様な機能があることを想定していない現状のMS-DOS等のシステムでも、FATキャッシュやセクタの開放を実現できた方が良いことは言うまでもない。

【0199】FATシステムは論理セクタ番号0に相当する部分にFATの場所やサイズといった情報を格納している。本実施形態では、このセクタを読むことでFAT

Tの場所やサイズを取得し、FATキャッシュの処理に利用する。同様に本来書き込みデータの内容を理解しないはずのICカードであるが、ファイルシステムの為の情報（ディレクトリエントリやFAT）を解析することでICカードが自立的に不要セクタを判断して開放する等の処理に役立てることが出来る。もちろんFATに限った話では無く、HFSやマッキントッシュ（商標）のファイルシステムでも書き込むデータの内容を解析すれば、不要セクタの検出が可能である。この様に構成することでATAハードディスクのインターフェースでも、ファイルシステムの動作に合わせた最適化処理を行うことを可能にする。

【0200】また、本発明は、複数の機器から構成されるシステムに適用しても、1つの機器からなる装置に適用してもよい。また、本発明はシステム或は装置にプログラムを供給することによって達成される場合にも適用できることは言うまでもない。この場合、本発明に係るプログラムを格納した記憶媒体が、本発明を構成することになる。そして、該記憶媒体からそのプログラムをシステム或は装置に読み出すことによって、そのシステム或は装置が、予め定められた仕方で動作する。

【0201】

【発明の効果】以上説明したように、本発明によれば、消去単位の大きいフラッシュROMをファイルシステムと適合させることを可能とした電子カメラが提供される。

【0202】

【図面の簡単な説明】

【図1】実施形態1におけるカメラシステムの構成を表すブロック図である。

【図2】本実施形態の電子カメラにおけるファイルシステムの階層構造を表す図である。

【図3】デバイスドライバの管理ブロックをC言語で記述した宣言文を示す図である。

【図4】フラッシュROM上のセクタ構造の例を示す図である。

【図5】管理領域用データと、データ領域とを分離して格納する構成を表す図である。

【図6】各フラグの状態に対応した意味を示す図である。

【図7】フラッシュROMにおけるセクタの書き換え手順を説明する図である。

【図8】本実施形態におけるフラッシュROMのガベージコレクション動作を説明する図である。

【図9】未使用セクタが存在しない場合のガベージコレクションの動作を説明する図である。

【図10】DRAM上に作成された記憶場所管理テーブルを説明する図である。

【図11】キャッシュソフトウェアの階層的な位置付けを表す図である。

【図12】キャッシュの主記憶上のデータ構造を表わす図である。

【図13】FATキャッシュの読み出し手順を表すフローチャートである。

【図14】FATキャッシュの書き込み手順を表すフローチャートである。

【図15】データ書き込み完了を確認するための動作手順をC言語で表現した図である。

【図16】DC/DCコンバータの出力容量を越えない様に電源を管理するプログラムをC言語で表現した図である。

【図17】本実施形態のリブートからサービスの開始までの動作手順を表わすフローチャートである。

【図18】指定セクタの読み出しサービスの手順を表わすフローチャートである。

【図19】論理セクタの書き込みサービスの手順を表わすフローチャートである。

【図20】主記憶上に獲得した退避データリストの様子である。

【図21】記憶を破棄する手順を表わすフローチャートである。

【図22】記憶効率の評価手順を表わすフローチャートである。

【図23】フラッシュROMの記憶領域の獲得手順を表わすフローチャートである。

【図24】ガベージコレクションの手順を表わすフローチャートである。

【図25】整理対象ブロック選出する手順を表わすフローチャートである。

【図26】整理対象ブロックの未使用セクタを使用済み化する手順を表わすフローチャートである。

【図27】整理対象ブロックの使用中セクタの移動手順を表わすフローチャートである。

【図28】整理対象となったイレースブロックの消去手順を表わすフローチャートである。

【図29】論理セクタの解放手順を表すフローチャートである。

【図30】実施形態2におけるICカードの構成を表すブロック図である。

【図31】本実施形態2のICカードを利用する為のホストシステムの簡単なブロック図である。

【図32】図31のホストシステムがICカードを接続する際の手順を示すフローチャートである。

【図33】ICカード内のマイクロコンピュータのメイシンシーケンスを示すフローチャートである。

【図34】ICカード内のマイクロコンピュータの割り込み処理の手順を表すフローチャートである。

【図35】I/Oアドレスの割り付け状態を表す図である。

【図36】電源種別に基づいてガベージコレクション処

理を切り換える場合の処理手順を説明するフローチャートである。

【図37】ファイルシステムよりファイル消去が指示された場合、不要セクタの解放手順を表すフローチャートである。

【図38】本実施形態における消去処理速度向上のための前処理の制御手順を表すフローチャートである。

【図39】本実施形態におけるフラッシュROMへの1バイトデータの書き込み手順を表すフローチャートである。

【図40】電源の共有手順を説明するためのフローチャートである。

【図41】本実施形態におけるフラッシュROMへの制御プログラムの格納状態を説明する図である。

【図42】相対アドレスで表現されたプログラムコード

の一例を表す図である。

【図43】図42のリロケーション情報レコードのデータを格納するテーブルを表す図である。

【図44】図41のプログラムを主記憶の8710番地へマッピングした場合のプログラムコードを示す図である。

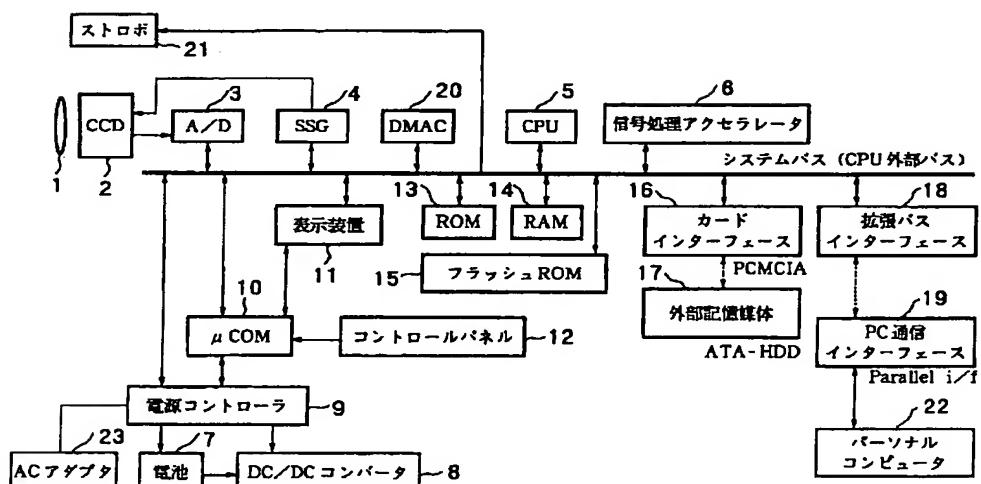
【図45】ディレクトリストロットの特徴を表す図である。

【図46】図45のディレクトリストロットにおいて、FileBが削除された状態を示す図である。

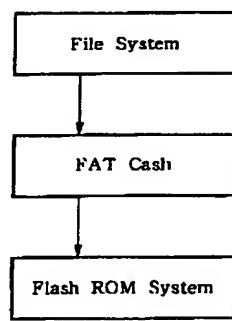
【図47】本実施形態のDOS互換ファイルシステムでファイルを消去した後の状態を表す図である。

【図48】本実施形態による電源資源(セマフォ)の時分割利用を説明するフローチャートである。

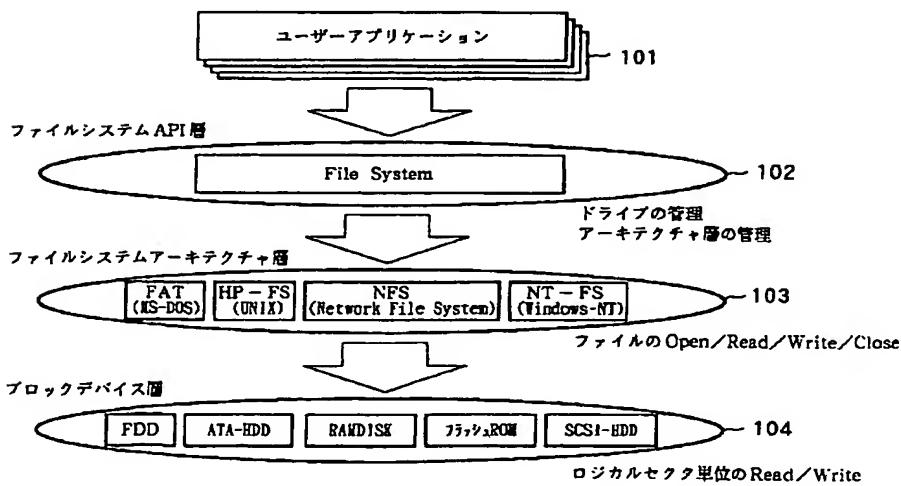
【図1】



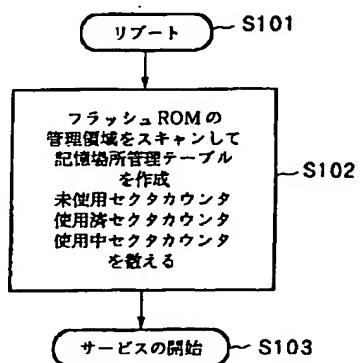
【図11】



【図2】



【図17】



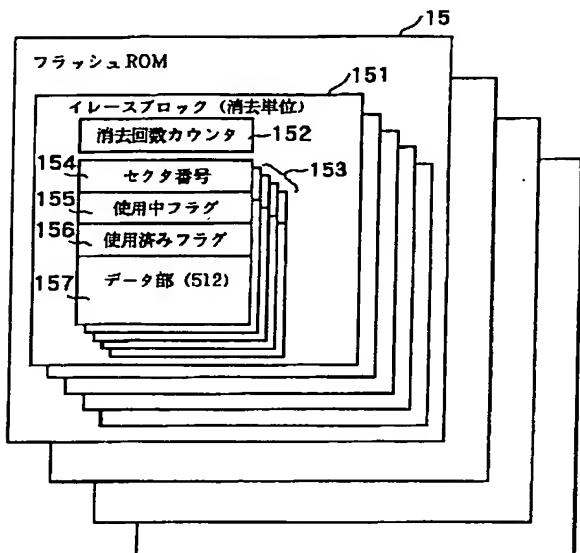
【図3】

```

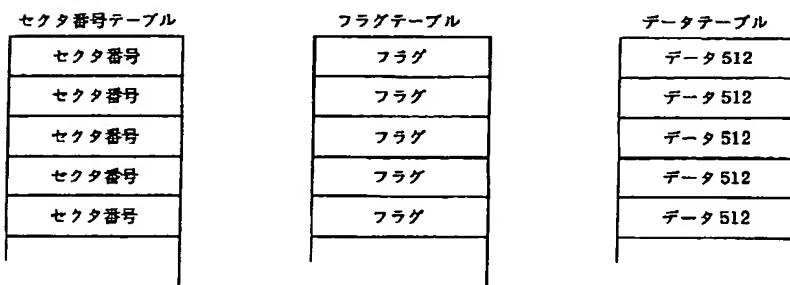
1: typedef struct DeviceTag {
2:     struct DeviceTag * Next :
3:     char DevName [32] :
4:     int (* InitDev) (void) :
5:     int (* ShutDown) (void) :
6:     int (* ReadSector) (long lsect, long nsect, char * buffer) :
7:     int (* WriteSector) (long lsect, long nsect, char * buffer) :
8:     int (* ReleaseSector) (long lsect, long nsect) :
9: } Device ;

```

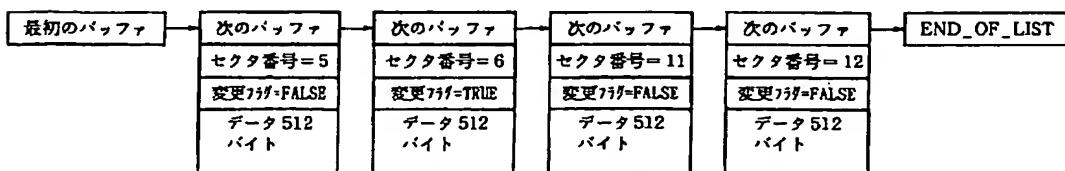
【図4】



【図5】



【図12】



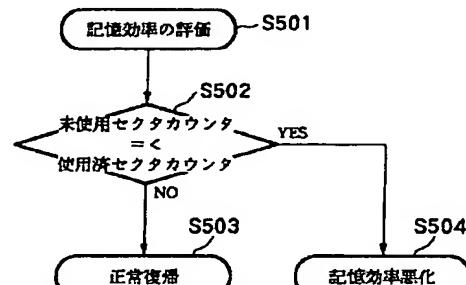
【図15】

```

1: void WriteByteEEP (struct DEV * Dev,char * Address,char Data)
2: {
3:     while (* Dev -> Address != Dev -> Data) :
4:         * Address = Data :
5:         Dev -> Address = Address :
6:         Dev -> Data = Data :
7:         RotateRdyQueue0 :
8: }
9: char ReadByteEEP (struct DEV * Dev,char * Address)
10: {
11:     if (Dev -> Address == Address) return Dev -> Data :
12:     while (* Dev -> Address != Dev -> Data) :
13:         return * Address :
14: }

```

【図22】

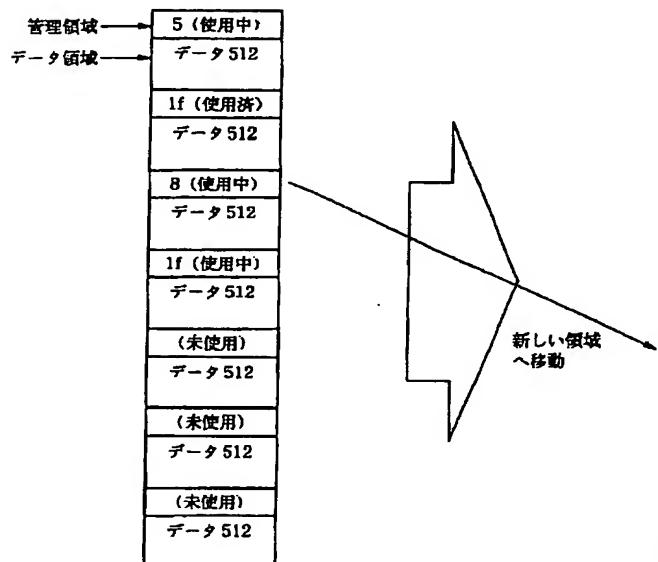


【図 6】

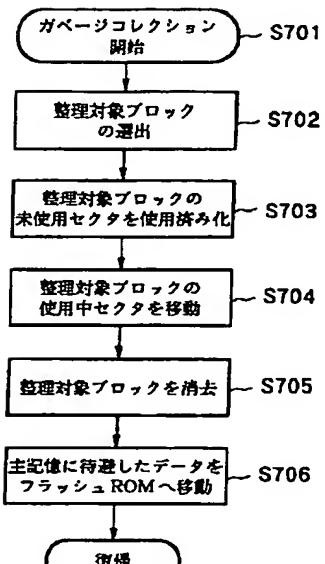
| 使用中フラグ | 使用済みフラグ | 意味 |
|--------|---------|---|
| FALSE | FALSE | 「未使用セクタ」...消去後の状態 |
| TRUE | FALSE | 「使用中セクタ」...セクタ番号が示すセクタをデータ部に格納している |
| TRUE | TRUE | 「使用済みセクタ」...データ部に格納されているデータは無効であり消去するまでこのセクタを利用できない |

FALSE : 消去後の状態 TRUE : FALSE のビット反転値

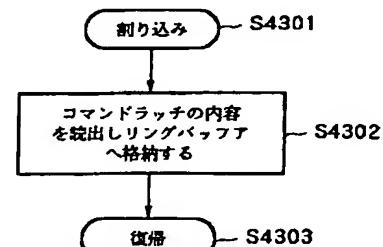
【図 7】



【図 24】



【図 34】

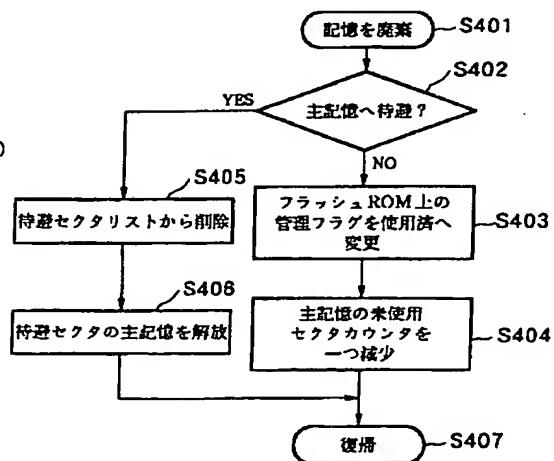


【図 16】

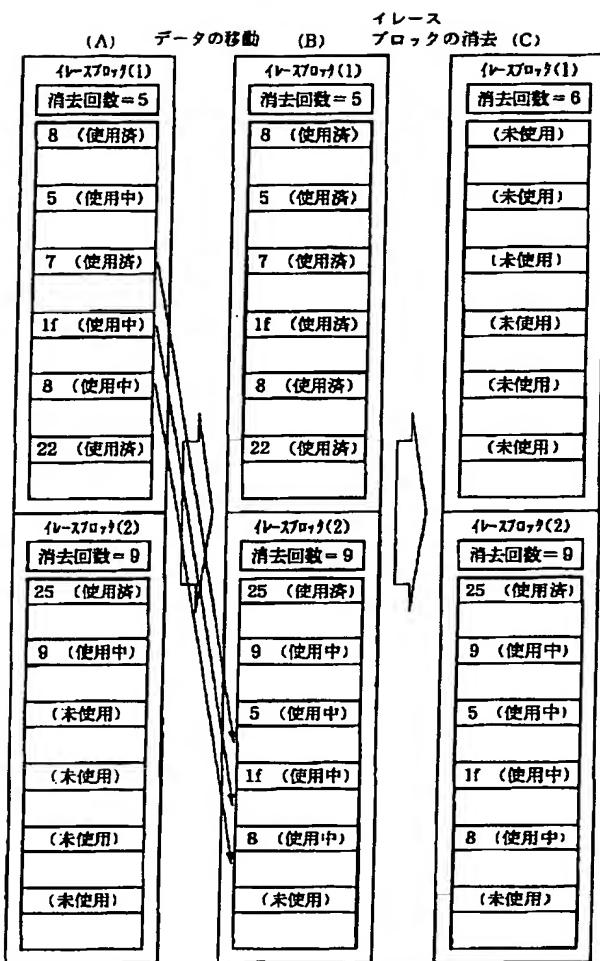
```

1: void ZoomUp (void)
2: {
3:     WaitSemaphore (SemDCDC):
4:     MotorStepUp (void):
5:     SignalSemaphore (SemDCDC):
6: }
7: char XWriteSectorEEP (struct * Dev, int Sector, char * Buffer)
8: {
9:     WaitSemaphore (SemDCDC):
10:    WriteSectorEEP (Dev, Sector, Buffer)
11:    While (* Dev -> Address != Dev -> Data):
12:    SignalSemaphore (SemDCDC):
13: }
  
```

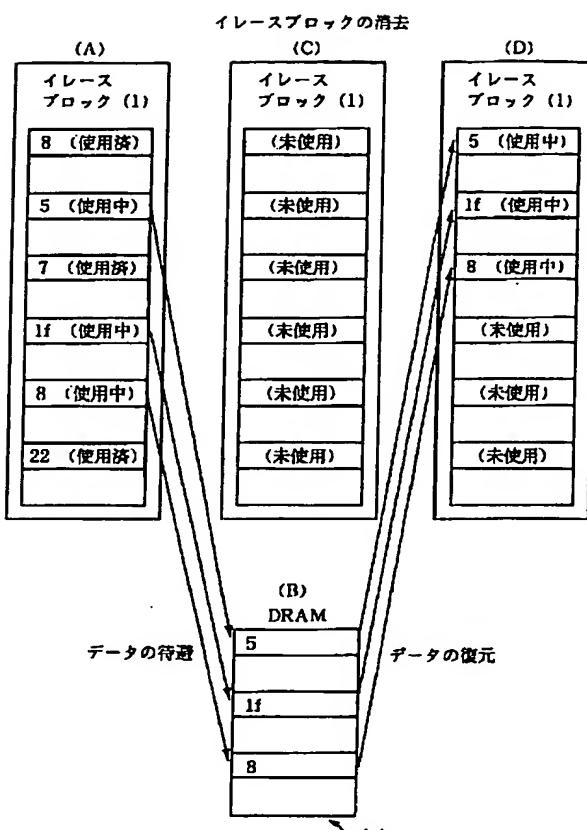
【図 21】



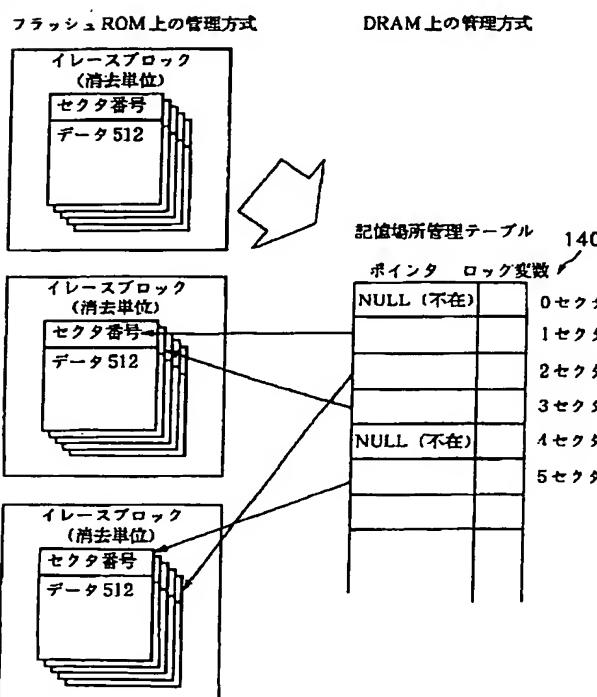
【図 8】



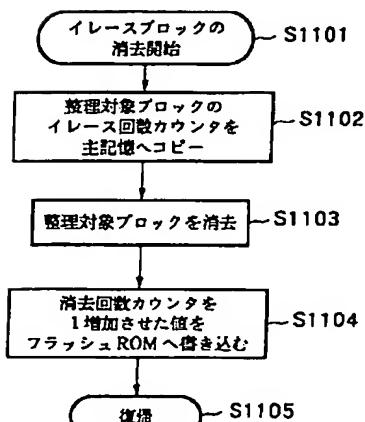
【図 9】



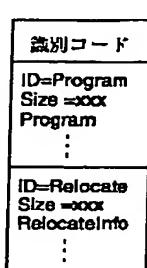
【図 10】



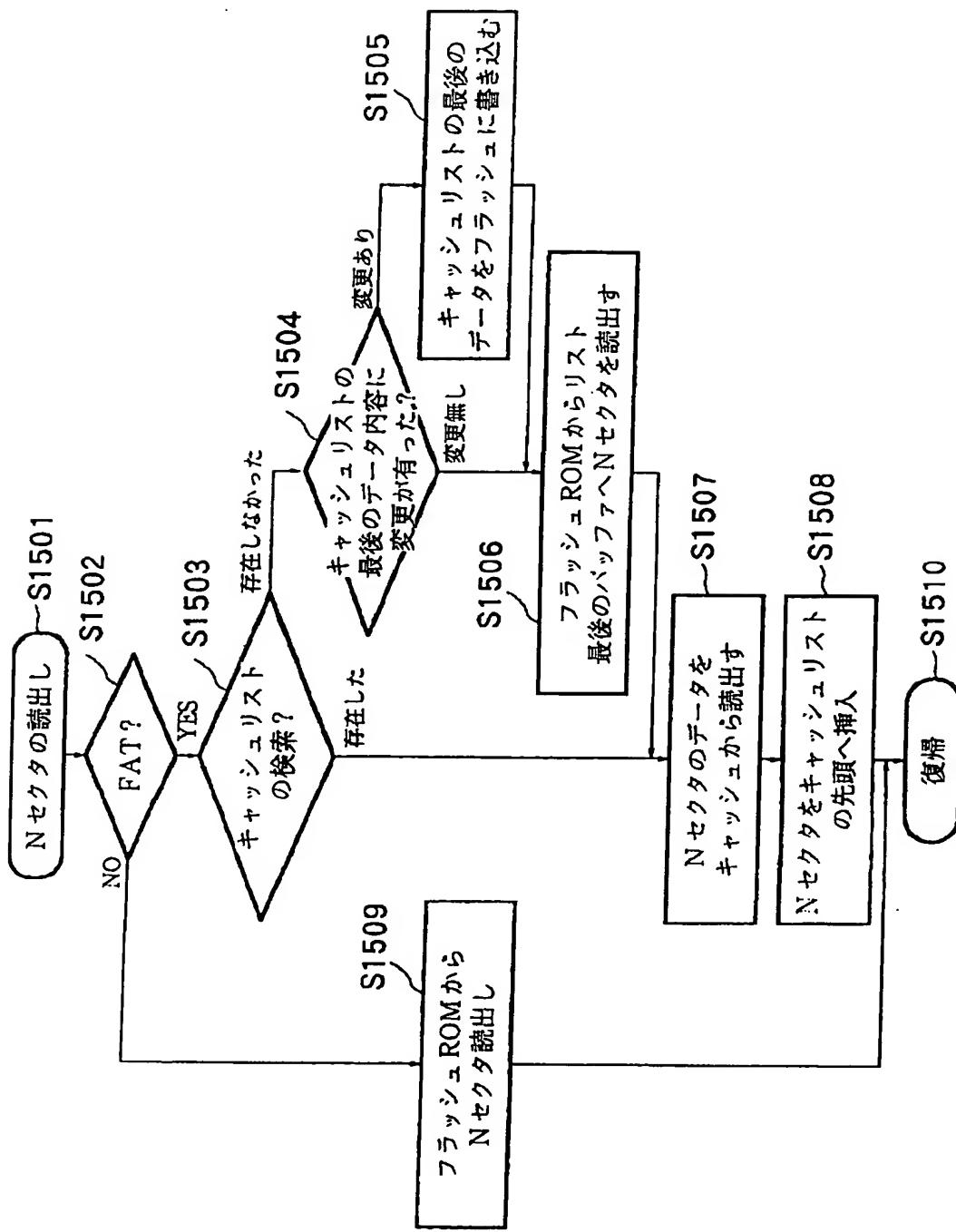
【図 28】



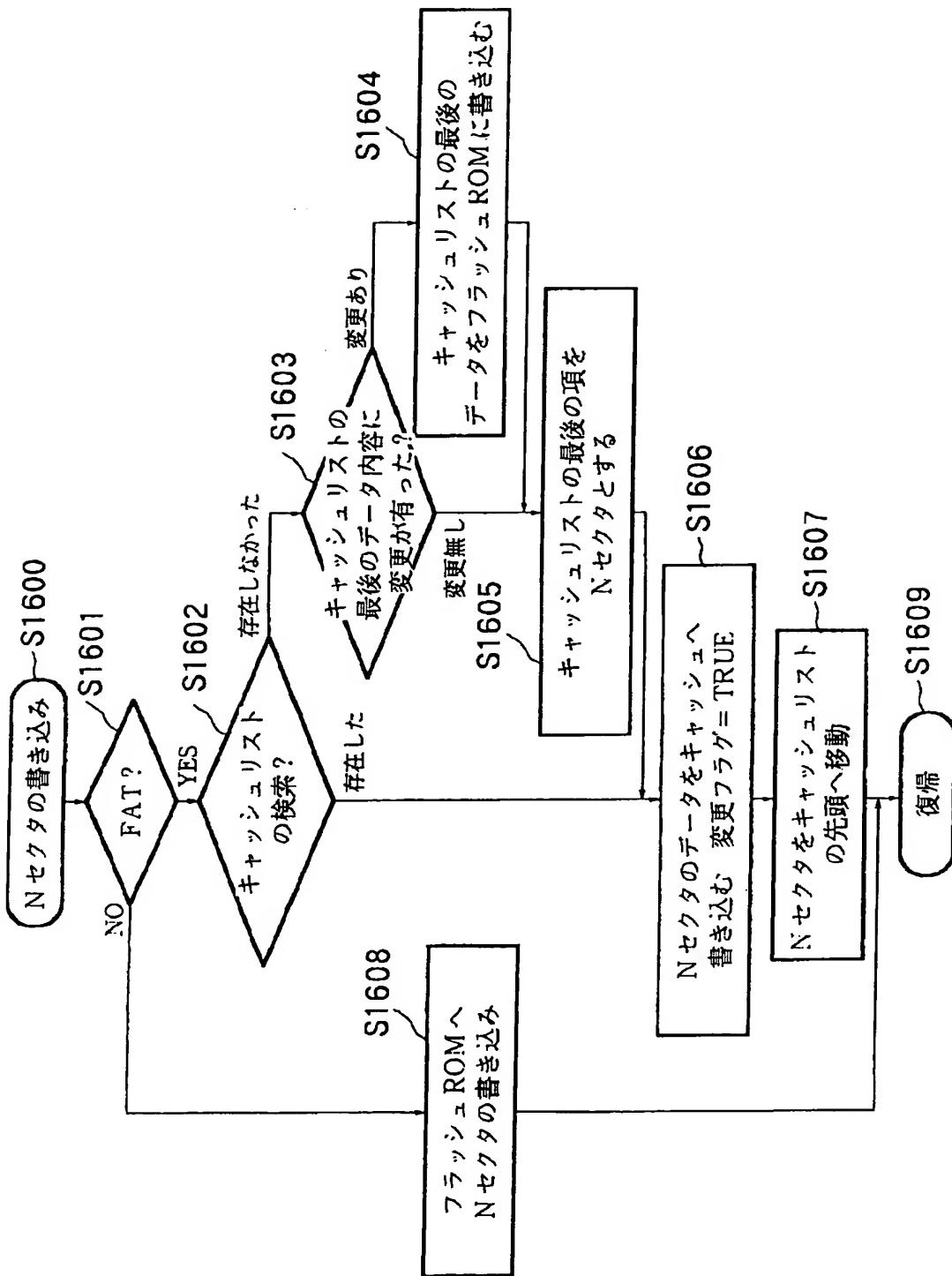
【図 41】



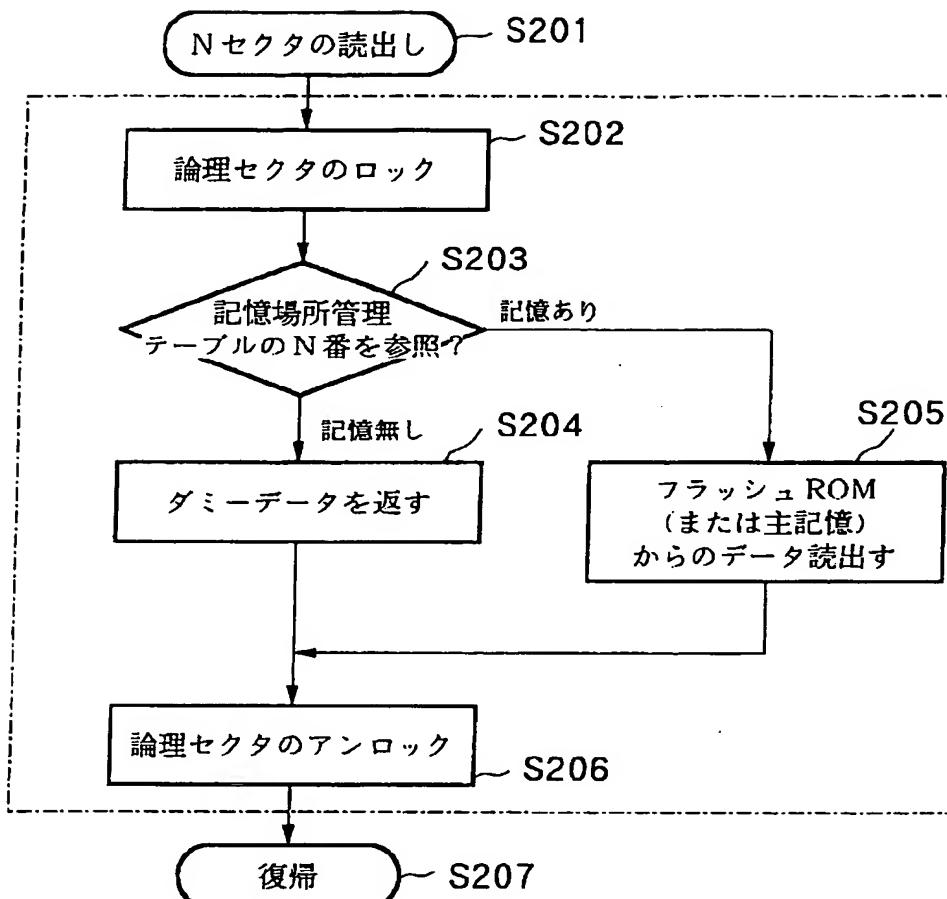
【図13】



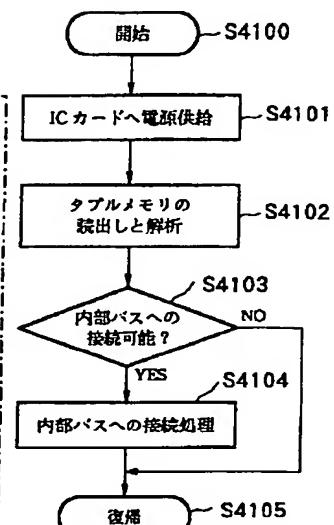
【図14】



【図18】

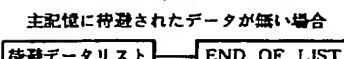


【図32】



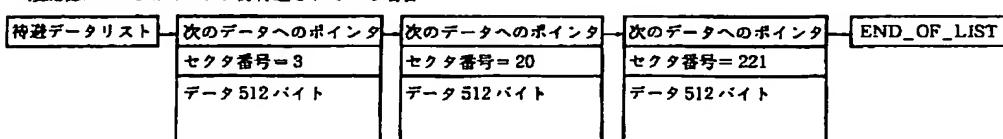
【図20】

(a)



(b)

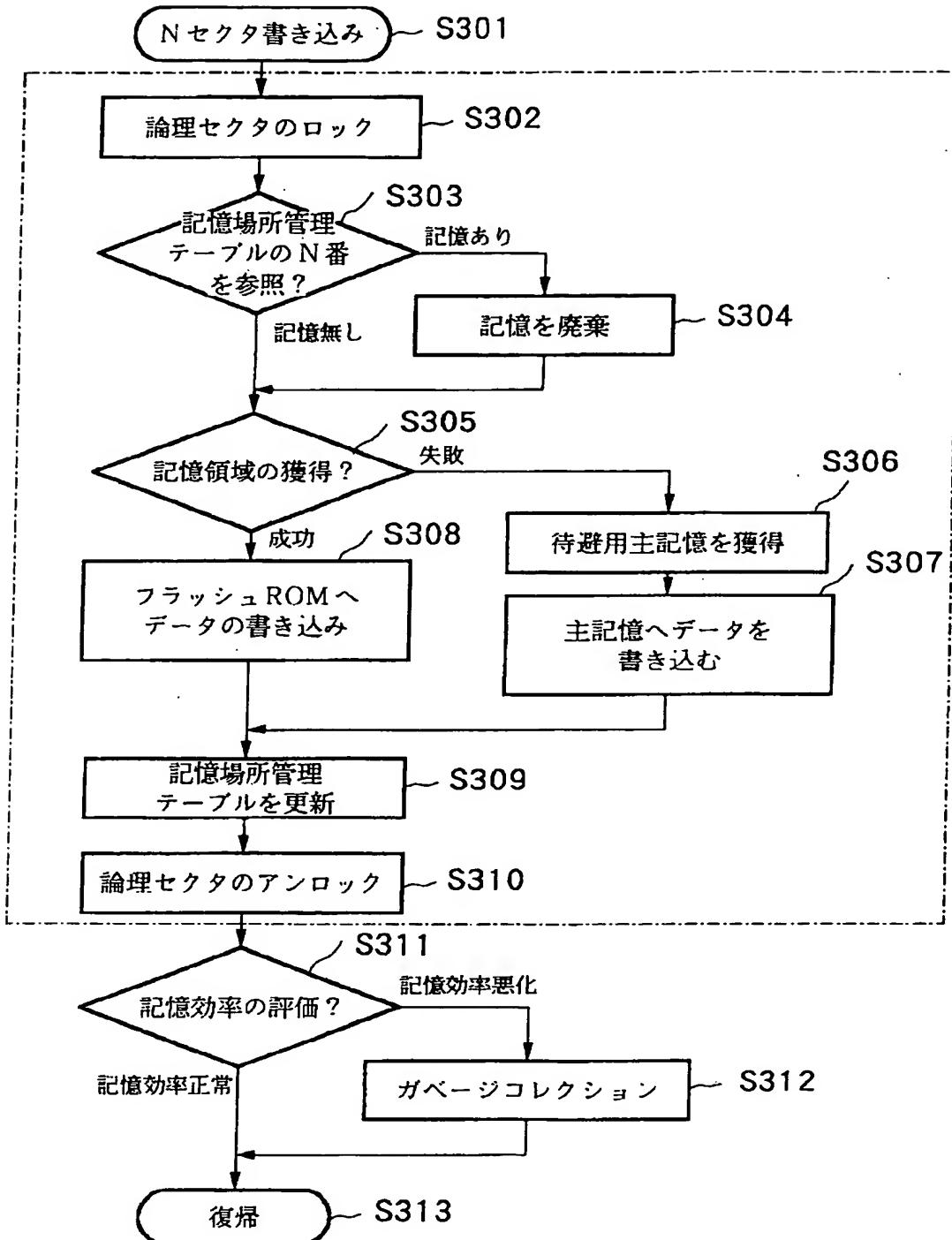
主記憶にデータが3セクタ分待避されている場合



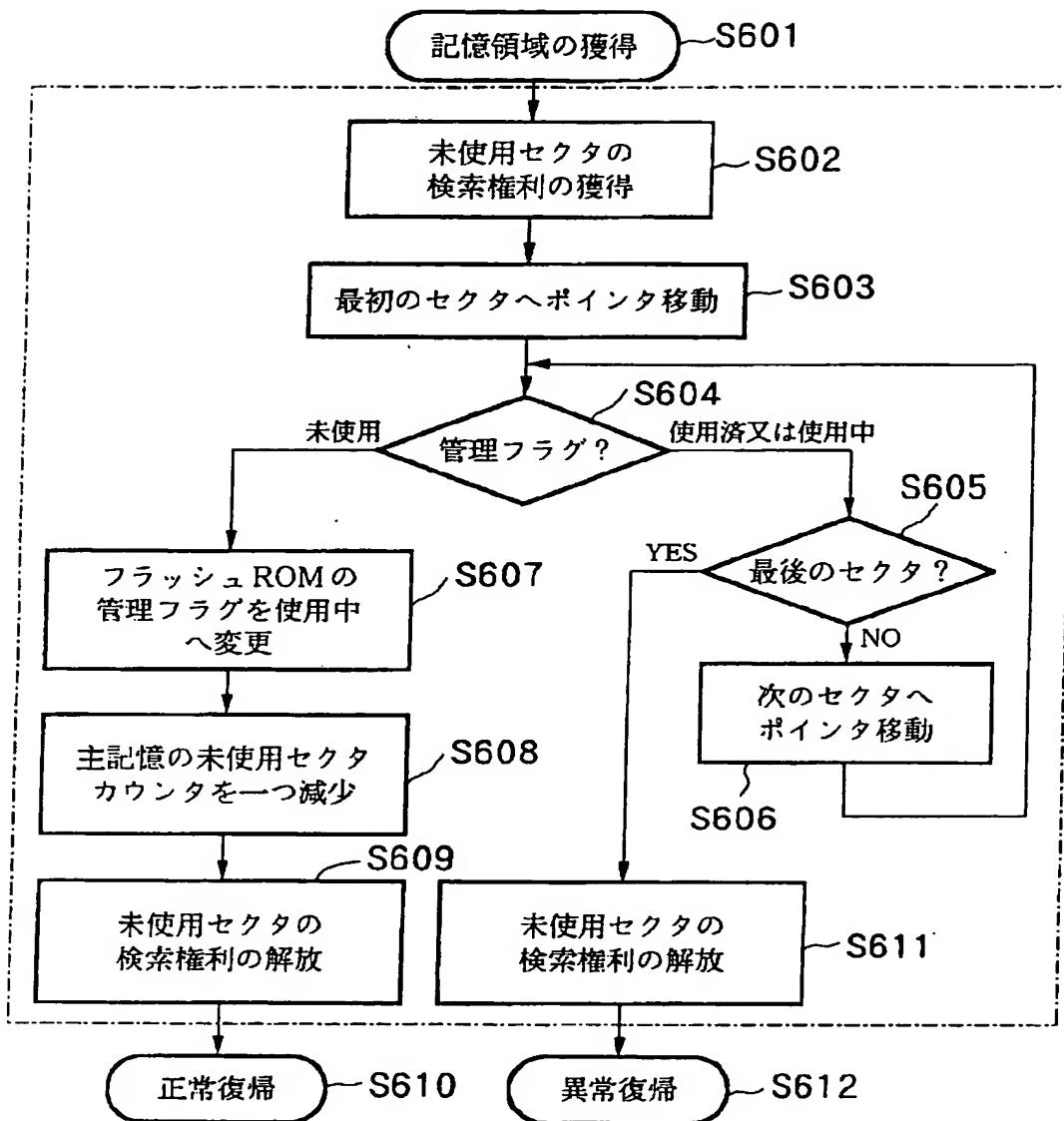
【図43】

プログラム中で絶対アドレスに変換しなければならない番地のテーブル
 0052
 0062
 ...

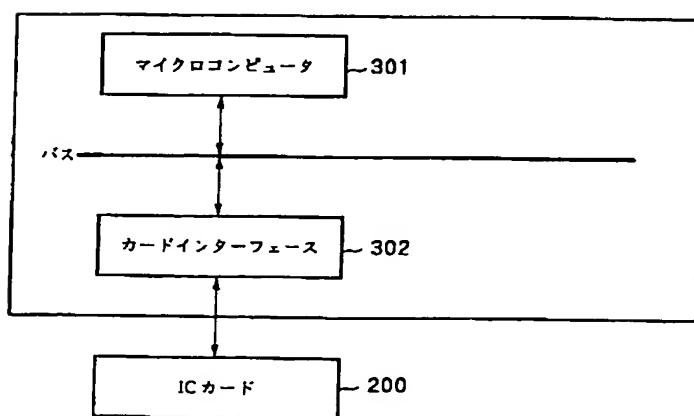
【図19】



【図23】



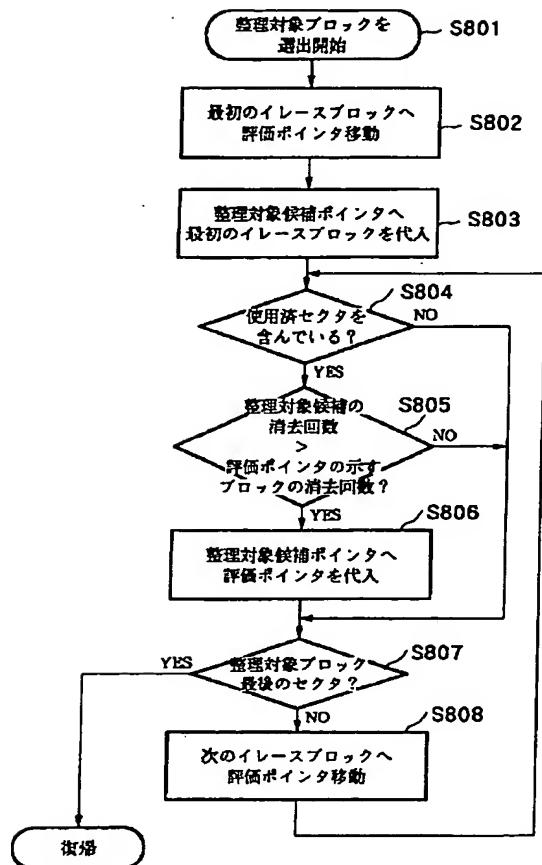
【図31】



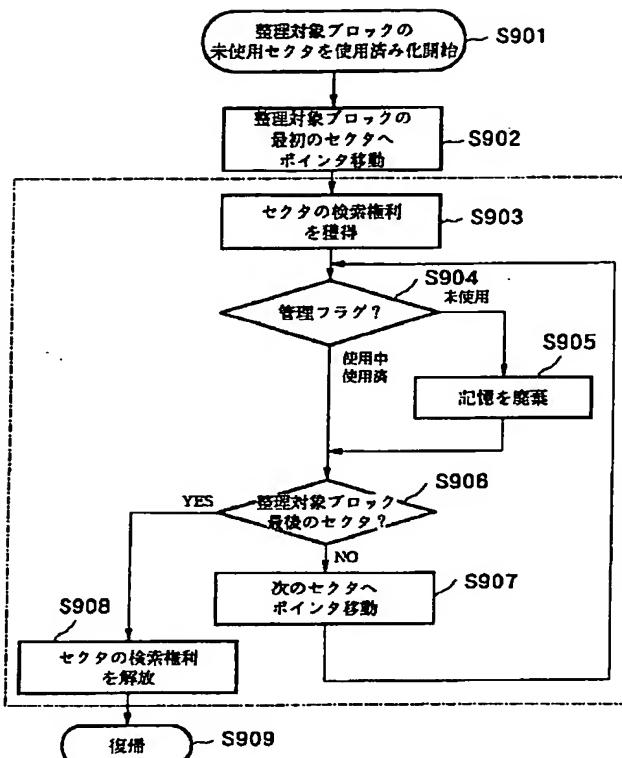
【図42】

| 相対アドレスで表現されたプログラムコード | | |
|----------------------|---------|--------------|
| 番地 | データ | ニーモニック |
| 0040 | 1234 | DB 1234 |
| 0050 | 85,0010 | JMP 0010 |
| ⋮ | | |
| 0060 | 86,-20 | MOV A,@(-20) |
| ⋮ | | |

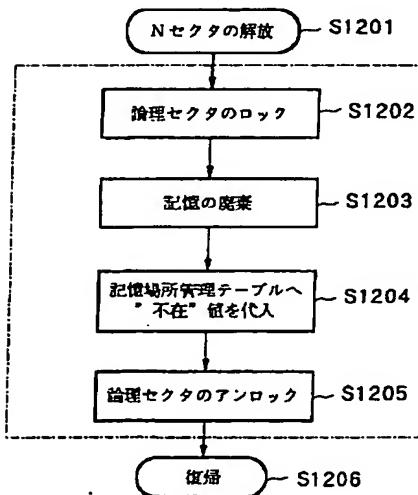
【図25】



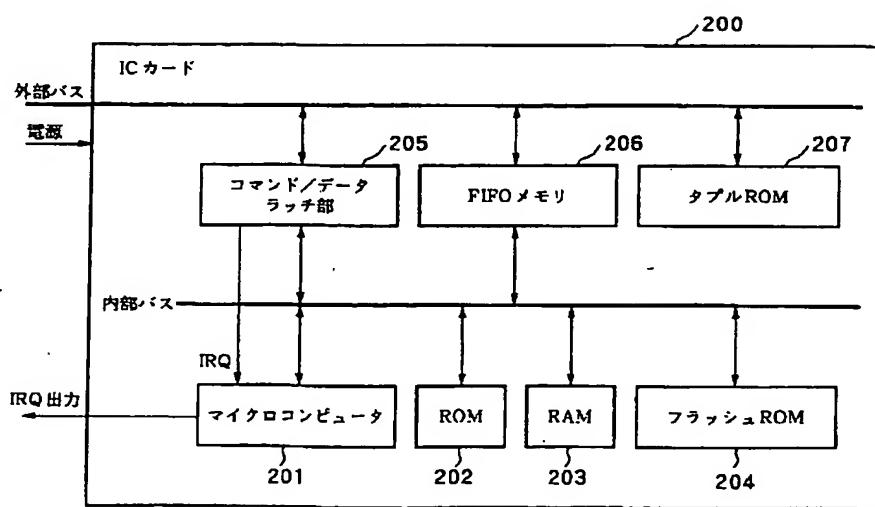
【図26】



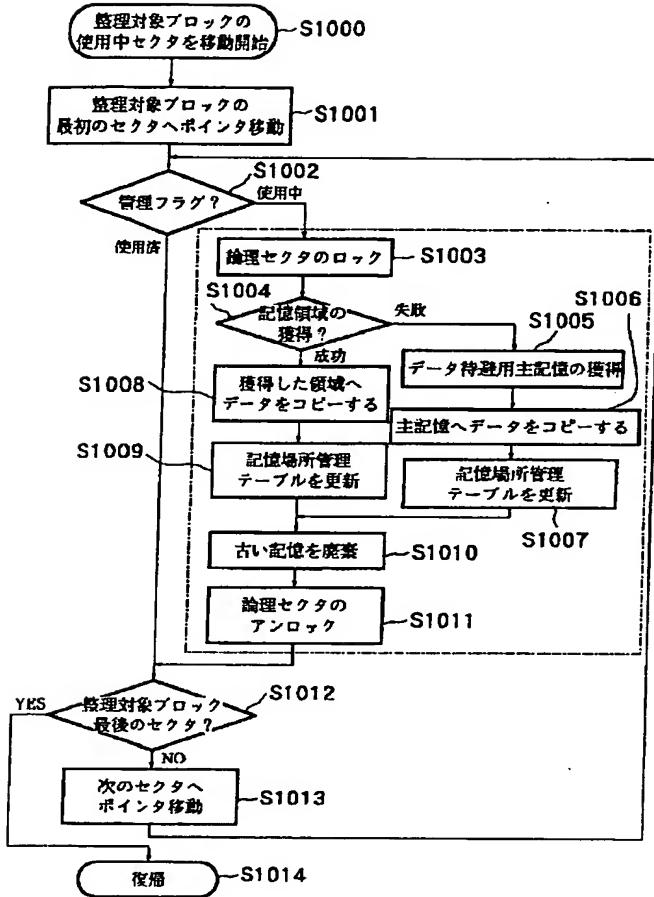
【図29】



【図30】



【図27】



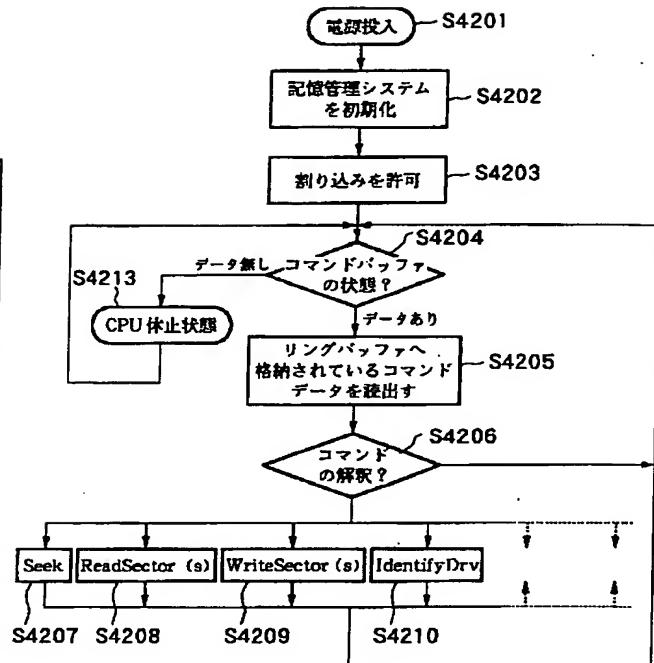
【図35】

| コマンド | |
|--------------|--------------|
| 読み出し (In) | 書き込み (Out) |
| Error | Features |
| SectorCount | SectorCount |
| SectorNumber | SectorNumber |
| CylinderLow | CylinderLow |
| CylinderHigh | CylinderHigh |
| Drive/Head | Drive/Head |
| Status | Command |

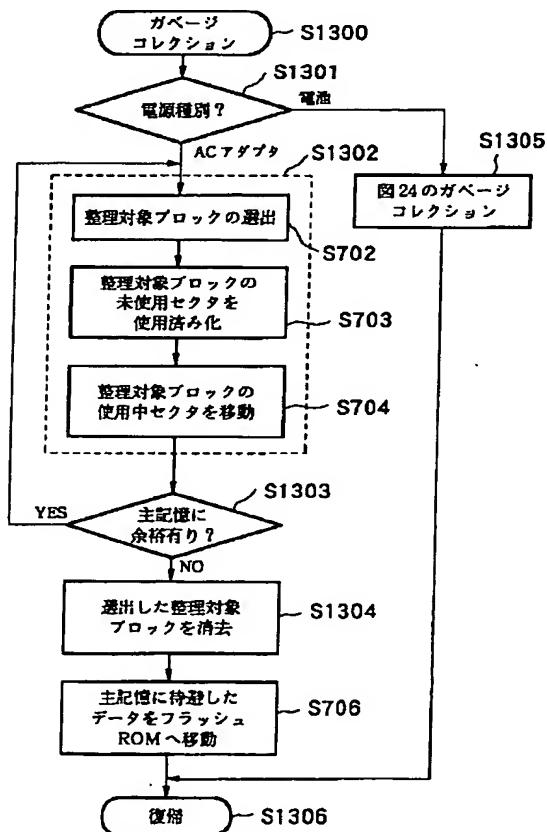
| コントロール | |
|------------|------------|
| 読み出し (In) | 書き込み (Out) |
| Alt.Status | Device Ctl |
| DriveAddr | |

割り込み発生

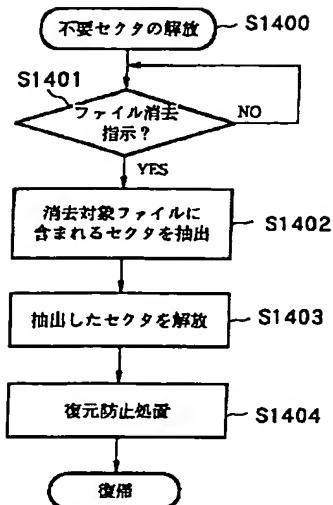
【図33】



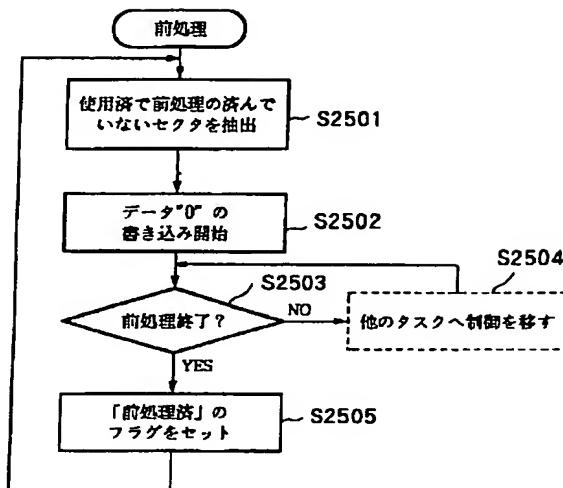
【図36】



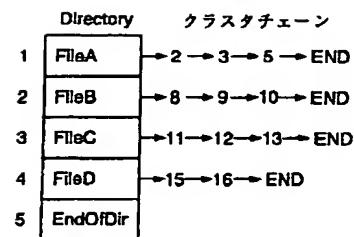
【図37】



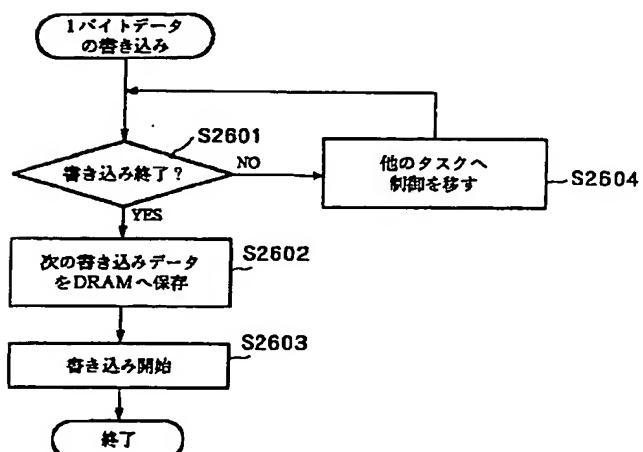
【図38】



【図45】



【図39】

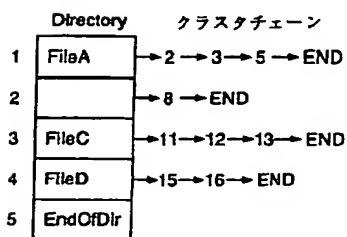


【図44】

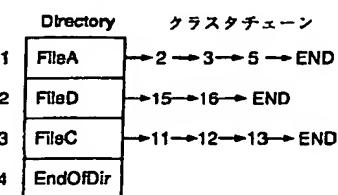
主記憶の8710番地へマッピングされたプログラムコード

| 番地 | データ | ニーモニック |
|------|---------|---------------|
| 8750 | 1234 | DB 1234 |
| 8760 | 85,8770 | JMP 8770 |
| ⋮ | ⋮ | ⋮ |
| 8770 | 88,8750 | MOV A,@(8750) |
| ⋮ | ⋮ | ⋮ |

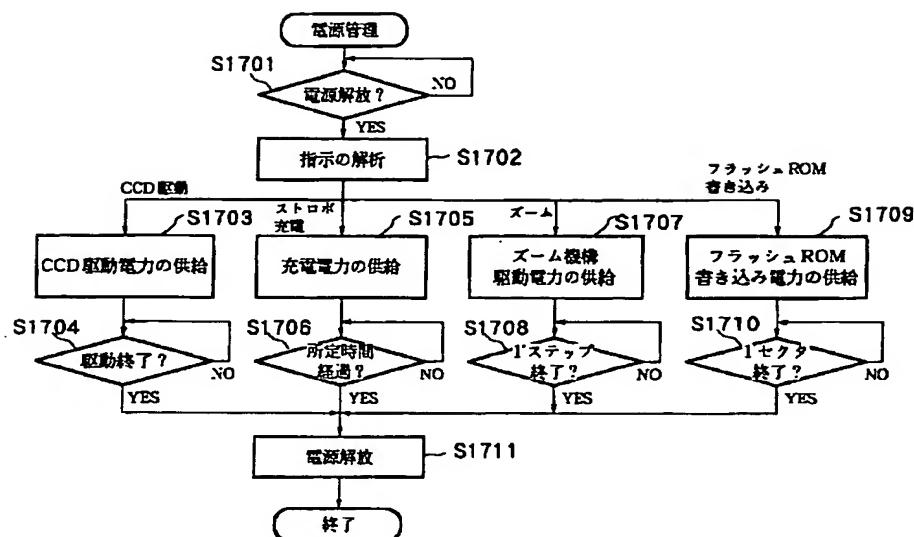
【図46】



【図47】



【図40】



【図48】

